

AD-A132 260

PROCESSOR-CONTROLLED DAC-20 (DIGITAL-TO-ANALOG
CONVERTER)(U) OKLAHOMA STATE UNIV STILLWATER
ELECTRONICS LAB W A HOLLOWAY ET AL. 15 MAY 83

1/1

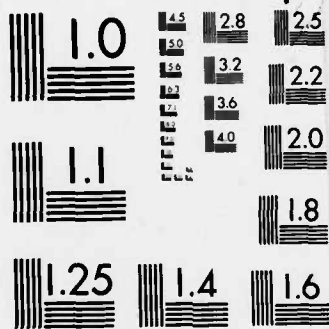
UNCLASSIFIED

SCIENTIFIC-3 AFGL-TR-83-0124

F/G 9/5

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12

AFGL-TR-83-0124

AD A 1 3 2 2 6 0

PROCESSOR-CONTROLLED DAC-20

W. A. Holloway and J. W. Spears

Electronics Laboratory - C.E.A.T.
Office of Engineering Research
Oklahoma State University
Stillwater, Oklahoma 74078

15 May 1983

Scientific Report No. 3

Approved for public release; distribution unlimited.

AIR FORCE GEOPHYSICS LABORATORY
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
HANSCOM AFB, MASSACHUSETTS 01731

DTIC
ELECTE
SEP 9 1983
S D

DTIC FILE COPY

83 09 08 017

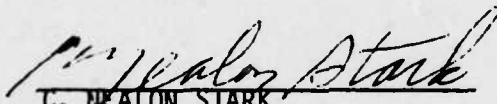
This report has been reviewed by the ESD Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS).

This technical report has been reviewed and is approved for publication


Contract Manager


EDWARD F. MCKENNA
Branch Chief

FOR THE COMMANDER


C. HEALON STARK
Division Director

Qualified requestors may obtain additional copies from the Defense Technical Information Center. All others should apply to the National Technical Information Service.

If your address has changed, or if you wish to be removed from the mailing list, or if the addressee is no longer employed by your organization, please notify AFGL/DAA, Hanscom AFB, MA 01731. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
AFGL-TR-83-0124	AD A132	260
4. TITLE (and Subtitle)		5. TYPE OF REPORT & PERIOD COVERED
PROCESSOR-CONTROLLED DAC-20		Scientific Report No. 3
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s)
W. A. Holloway and J. W. Spears		F19628-81-C-0079
9. PERFORMING ORGANIZATION NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Electronics Laboratory, C.E.A.T. Oklahoma State University Stillwater, Oklahoma 74078		62101F 7659048BB
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE
Air Force Geophysics Laboratory Hanscom AFB, Massachusetts 01731 Contract Manager/Jack R. Griffin/LCR		15 May 1983
		13. NUMBER OF PAGES
		76
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)		
Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
PCM Decoding, Digital-to-Analog Converters, Microprocessors, Ground Support Equipment, Digital Word Selection		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)		
<p>The Processor Controlled DAC is a PCM Word Selector, controlled by dual 6502B microprocessors. It was developed for a high word rate PCM systems, and as a flexible unit to be used with almost any PCM format. This multi-channel DAC (digital-to-analog converter) is capable of selecting up to 32 separate word patterns and can produce up to 30 outputs with additional DAC's. At present, 18 DAC outputs are offered with 0 to +10 volt outputs. Details of the circuit design, operation, and software of the "PROCESSOR-CONTROLLED DAC-20" are covered in this report.</p>		

DD FORM 1473

1 JAN 73

EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SUMMARY

The OSU Model D90PR01 processor-controlled DAC (digital-to-analog converter) was developed for use as a high rate, multi-channel Word Selector. The dual microprocessor controlled "DAC-20" will presently provide up to 18 separate DAC channels of real time or playback data through a wide variety of high word rate PCM systems. With the addition of more DAC's up to 30 data outputs may be produced. There are presently ten 10-bit DAC's, and eight 12-bit DAC's in the system. Each channel produces a 0 to +10 volt output.

The unit will accept up to 100 words per minor frame and 100 minor frames per major frame with a maximum word rate of 125 KHz. PCM words may be up to 16 bits in length. Each DAC in the unit may be programmed in the form of a selected word per major frame, word every minor frame, or a word by a frame or interval repeat sequence.

Each DAC may be calibrated by percentages of their full scale outputs. A 0, 25, 50, 75, and 100 per cent scale calibration may be manually stepped through, or calibrated automatically by processor control.

A computer interface can be developed to make use of all 32 possible selected words. This interface would not affect the operation of the DAC-20 unit, and would make it a "PCM sorter" for a computer. This interface could also allow a computer to externally program the DAC Processor.

This report is written to serve as a technical manual for the theory of operation of the unit.

Additional Notes:

19 July 83

Since preparation of the manuscript copy for this report, two additional 12-bit DAC channels have been added to the DAC-20, providing the full 20-output capability planned. The additional circuitry was installed on Card No. 5, and is shown in OSU Drawing D90PE07C. The parts list on page 34 does not reflect this addition.

R. F. Buck

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	



TABLE OF CONTENTS

Topic	Page
1.0 INTRODUCTION.....	1
2.0 GENERAL DESCRIPTION.....	3
2.1 Layout	3
2.2 Operational Procedures.....	3
2.2.1 Decom Interface.....	3
2.2.2 Initialize Mode.....	3
2.2.3 DAC Set Mode.....	5
2.2.4 System Test.....	6
2.2.5 Computer Interface.....	6
2.2.6 Limitations.....	6
3.0 HARDWARE.....	9
3.1 Introduction.....	9
3.2 Card 1- Processors.....	9
3.3 Card 2 - Interface Card.....	9
3.4 Card 3 - 10-Bit DAC's.....	11
3.5 Card 4 - 12-Bit DAC's.....	13
3.6 Card 5 - Update RAM Card.....	13
3.7 Front Panel.....	13
3.8 Rear Panel.....	13
4.0 SOFTWARE	15
4.1 Development.....	15
4.2 Data Entry and Calibration.....	18
4.3 Compiling the Data Entered.....	18
5.0 OPERATIONAL PROCEDURES.....	27
6.0 PCM DECOM INTERFACE.....	29
7.0 PARTS LIST.....	31
APPENDIX.....	35

LIST OF ILLUSTRATIONS

Figure No.	Title	Page
1	Processor Controlled DAC-20.....	2
2	Typical System, Block Diagram.....	4
3	Common Memory Timing.....	8
4	ID Detection Timing.....	10
5	Block Diagram, DAC-20 Hardware.....	12
6	View of Rear Panel, DAC-20.....	14
7	DATA Entry Flow Chart.....	16
8	Flow Chart for WDACUP.....	20
9	Processor 1 Flow Chart.....	37
10	SET Flow Chart (2 pages).....	43
11	POWER Flow Chart (2 pages).....	47
12	NUM Flow Chart.....	52
13	CALIBRATION Flow Chart (2 pages).....	54
14	KEYBD Flow Chart.....	58

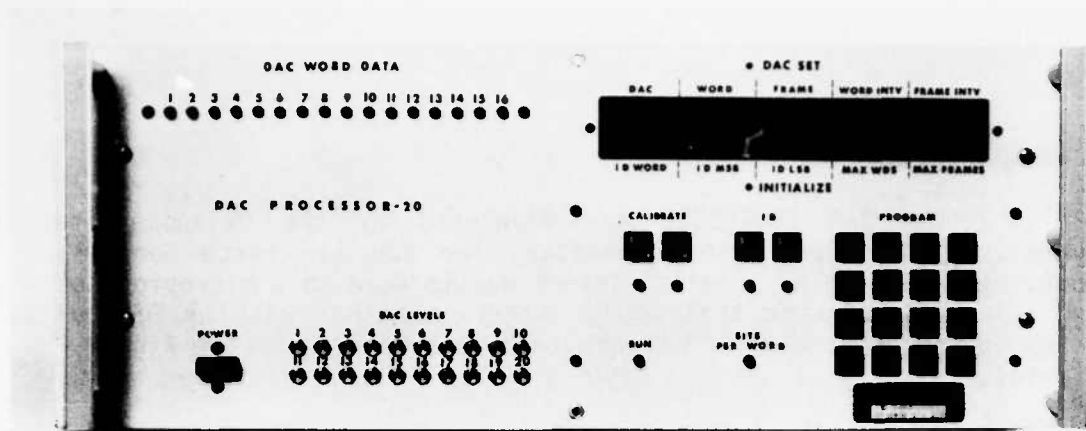
1.0 INTRODUCTION

1.1 The DAC PROCESSOR was developed by the Oklahoma State University (OSU) Electronics Laboratory for the Air Force Geophysics Laboratory (AFGL). The original intent was to develop a microprocessor-based PCM word selector that would interface either with an OSU-built (or any other compatible) PCM decommutator and operate at the higher PCM bit rates.

1.2 OSU PCM decommutator (decom) development began in 1959 for AFGL, for devices to be used in the laboratory or in the field. Later, some of these decons had as many as five low resolution (8-bit) analog outputs with parallel printer and computer interfaces. Increased requirements for preflight checks and real time data display, as well as analog recording, dictated the development of multiple channel digital-to-analog systems. A 16-channel DAC (digital-to-analog converter) unit (Model C90DA01) was developed to interface with pen recorders and bargraph displays. The unit had 8-bit resolution for its analog outputs. An 8-channel DAC (Model C90DB05) was designed that provided 10-bit DAC's and a 12-bit light display for the selected words. These two units were constructed under contract to AFGL.

1.3 The unit in this report was developed for more resolution and greater flexibility. The resolution was accomplished with 10- and 12-bit DAC's, with a very low level of noise on the outputs (less than 2 mv in a 10 volt range). (Particular attention was paid to grounding.) The flexibility of the unit was increased by using microprocessors to select the PCM data. Dual 6502B microprocessors, communicating through a common 1K of memory, resulted in a software-controlled, easily expandable word selector.

1.4 The DAC PROCESSOR was originally developed for the AFGL FIRSSE sounding rocket project, which has a high bit rate PCM. The flexibility of the unit allows it to be used with almost any PCM format, making it a general purpose device. Future expansion of the number of output DAC channels (beyond the initial eighteen outputs provided in this unit) is presently planned. This will further increase utility of this device.



Front Panel
Top View, Cover Open

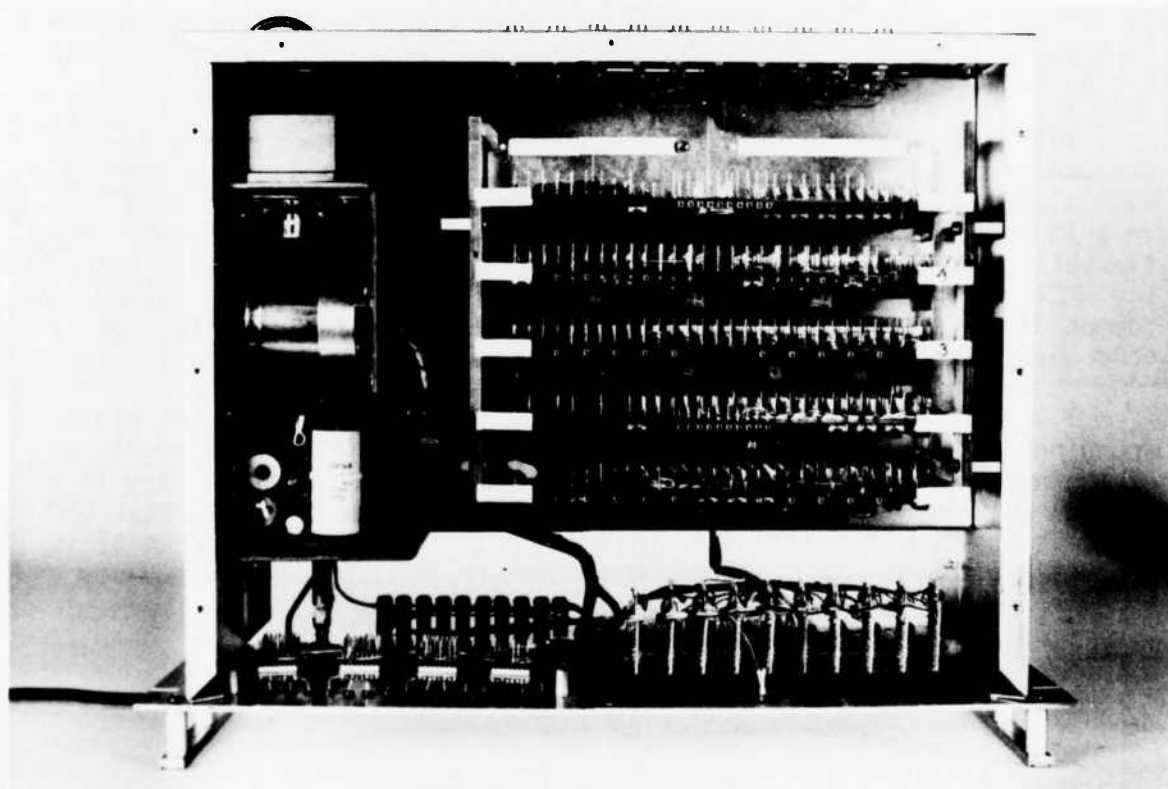


Figure 1. Processor Controlled DAC-20 Views

2.0 GENERAL DESCRIPTION

2.1 LAYOUT

2.1.1 The DAC Processor is housed in a rack-mountable Optima cabinet with handles. The cabinet measures 17x13 1/2x7 inches with a 7x19 inch front panel. The DAC Processor unit weighs 25 pounds.

2.1.2 The front panel consists of a programming keypad and a 10-digit decimal LED display. The keypad is used to initialize the system and to enter data. The 10-digit display monitors what has been entered. Other front panel features include the ID select (BCD or BIN), calibrate select (MANual or AUTO), a 16-bit LED display (for showing the digital display of a selected word), DAC level set potentiometers, and the power ON/OFF switch. These features are visible in the photograph of Figure 1.

2.1.3 On the back panel are 20 BNC connectors for the DAC outputs, a 50 pin CINCH type connector for decom interface, fuse, and power cord. Fan and air inlet ports are also located on the back panel. These features are shown in Figure 6, on page 14.

2.1.4 The chassis inside the DAC Processor cabinet includes two power supplies, five plug-in wire-wrapped cards, and the 10-digit LED display card (which is mounted on the back of the front panel).

2.1.4.1 The power supplies incorporated are a +5 volt supply and a ± 15 volt supply. TO-220 type regulators tap the supplies to provide -5 volt and +12 volt power for the EPROMS.

2.1.4.2 A typical usage of the unit is depicted by the system block diagram of Figure 2.

2.2 OPERATIONAL PROCEDURES

2.2.1 DECOM - The DAC Processor is interfaced to the decom through one cable with a 50 pin CINCH type connector on each end. Through this cable, the decom must provide a positive word clock, eight BCD word address lines, and sixteen PCM data lines. The decom must be in the "All Words" position.

2.2.2 INITIALIZE MODE - When power is turned on to the DAC-20 system, all registers and counters are reset, and all DAC information is "zeroed". The DAC Processor is automatically set to the INITIALize mode. In this mode the ID, minor frame and major frame sync information, and bits per word are entered.

The ID in the PCM data is used for subcom word selection. The ID should follow the frame sync word(s) and be within one PCM word. The ID word number location and it's MSB (most significant bit) and LSB (least significant bit) must be entered to initialize the ID shift circuitry. The ID may be in any word, and be at least 4 bits wide. The ID is automatically initialized at Word 1, but the word number may be changed by moving the cursor to the left and entering the word number the ID is

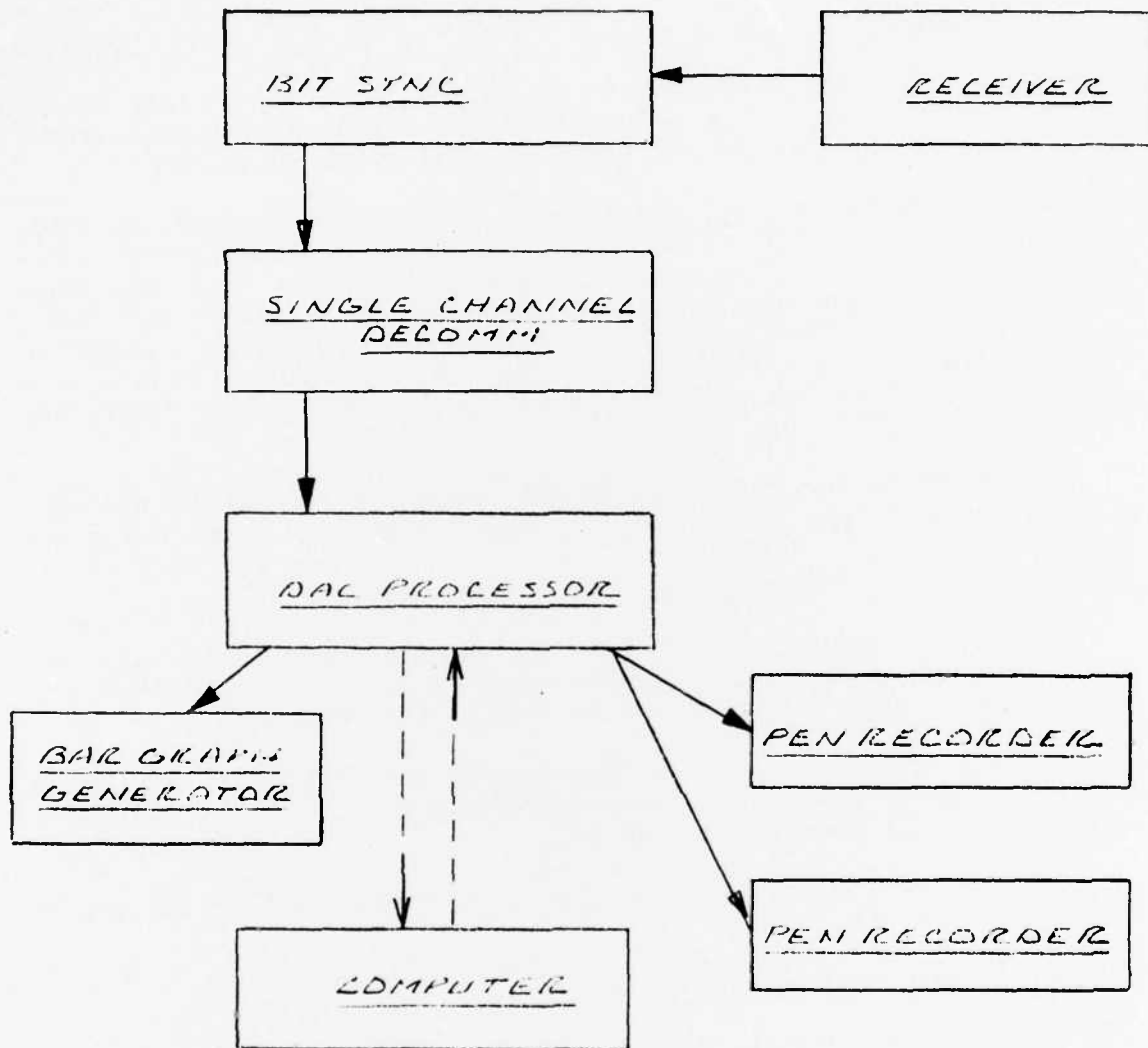


Figure 2. Typical System Block Diagram

in. The unit assumes an incrementing ID; therefore, the MSB is the first bit past any fixed bits. The LSB is the last bit in the ID.

EXAMPLE: Assume an 8-bit ID in WORD 1, 100XXXX (7 bits long). The first three bits are fixed. Therefore, the MSB to be entered is 4, and the LSB to be entered is 7.
BCD or BINary ID is selected from the front panel pushbuttons.

The number of maximum words and frames entered initialize the software counters which enable the DAC's. The maximum number of words is the total number of words in the minor frame. The minimum number of words that can be entered is 3 and the maximum number of words the DAC-20 system is capable of handling is 100 (enter 00 for 100 words).

The maximum number of frames is the total number of minor frames in a major frame. The limits are a minimum of one frame and a maximum of one hundred frames (enter 00 for 100 frames). The word rate should not be greater than 125 KHz (10 μ s per word) and PCM words may be up to 16 bits in length.

Processor number 2 in the unit assumes there are 16 bits per word unless a change is made. The STEP key may be depressed and the correct number of bits per word may be entered. The processor will turn off the unused bits. The BITS/WORD LED will come on during this entry.

In the INITIALize mode, the DAC's may be calibrated by the percentages of their full scale output. These percentages are 0, 25, 50, 75, and 100 per cent. The calibration may be manually stepped through by pressing the MAN key at the operator's convenience. The unit will remain in the MANUAL mode until the SET or INIT keys are depressed. The DAC's may also be calibrated automatically by pressing the AUTO switch. By processor control, the unit will calibrate the DAC's from 0, stepping through to 100 percent and back to 0, and return the unit to the INITIALize mode. Pressing the RUN button returns the unit to normal operation.

The DAC outputs on the rear of the panel have a 0 to +10 volt range with approximately 1.5 millivolt noise level. Level adjustments for each DAC are located on the front panel. Zero offset adjustments are located on each DAC card. There are ten 10-bit DAC's (1-10), and eight 12-bit DAC's (11-18) into which selected words may be latched. The outputs of the DAC channels are capable of driving pen recorders, bargraph displays, and most monitoring devices. Up to 32 different word patterns may be programmed into the DAC processor. However, since only 18 DAC's are presently installed, only those words programmed in the first 18 DAC positions will have an output. Depending on the physical size of the DAC's, up to 30 DAC's may be installed in the DAC Processor.

2.2.3 DAC SET MODE - The SET mode is used to initialize the DAC's with selected word, frame, word interval, and frame interval information. When the set mode is initiated, the DAC SET LED will come on and DAC 1 will be displayed, ready for the selected word information to

be entered. Pushing the STEP key will increment to the DAC 2 position for selected word entry. (Note: Holding the STEP key in will cause the DAC numbers to continue to increment.) This process is repeated until all desired data is entered.

To change word information in a particular DAC channel, the cursor (decimal point on the display) may be moved to the DAC position and the desired DAC number entered. Remember that holding the STEP key in will increment the DAC numbers. (This works only in the DAC SET mode.)

An important point to mention is that no two DAC's can have identical data. If the same data is entered in two DAC's, the second entry will be ignored.

Once all information is entered, the RUN key is depressed. This mode sets the PCM data and latches in the DAC's and Word Display. The last DAC information that was entered in the SET mode will be latched into the Word Display. If, while in the RUN mode, the SET or INIT keys are pressed, the DAC's will not be updated until the RUN key is again pressed.

2.2.4 SYSTEM TEST - The simplest test that may be performed to check the operation of the system is to calibrate the DAC's, manually or in the AUTO mode. Processor 1 latches all the DAC's and the word display with the binary data. Processor 2 reads the keyboard and displays the calibration percentage on the 16-bit LED display.

If either function isn't performed, the unit can be turned off and on to reset. If, while in the RUN mode, the DAC's are not being updated, the processor may not be able to obtain the proper subframe ID. The initialization data should be checked to insure it corresponds to the correct PCM format.

2.2.5 COMPUTER INTERFACE - A computer interface may be developed to make use of all 32 possible selected words. This would not affect the operation of the DAC-20 system, but would allow it to be a PCM data sorter for a computer. Another aspect of an interface is that the computer could externally program the DAC Processor.

2.2.6 LIMITATIONS - A few limitations must be noted in use of the processor-controlled DAC in the present configuration. They may be summarized as follows:

1. With processors operating at the present rate of 2 Mbit/sec, the minimum duration for a word within the PCM data string is 10 microseconds (e.g., maximum rate is 2 Mbit/sec for 20-bit word length).
2. Although software and programming provide for a maximum of 32 selected words, the current hardware includes only 18 digital-to-analog conversion circuits. As a result, only DAC numbers 1 through 18 may be programmed to supply an output from the rear panel.

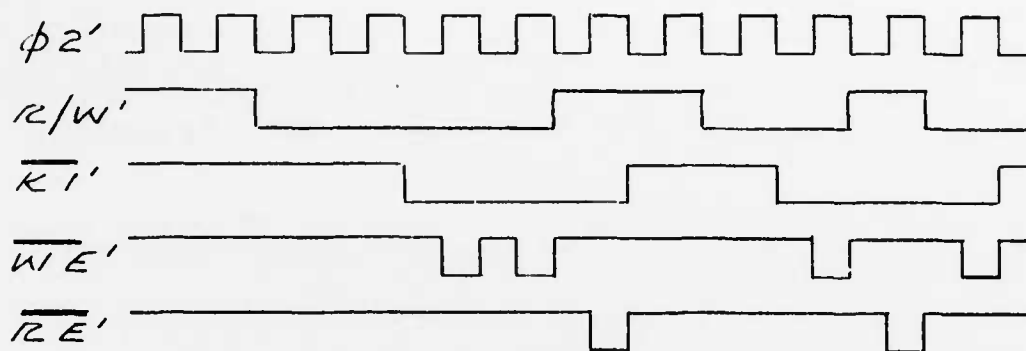
3. A selected word cannot be entered on more than one DAC. If two DAC's are programmed for the same word information, the lowest DAC number will have priority and the higher number(s) will be inhibited.

4. In programming for "supercomm" repeated data, if the same word is entered for "Word Interval" and "Frame Interval" selection, the Frame Interval program has priority.

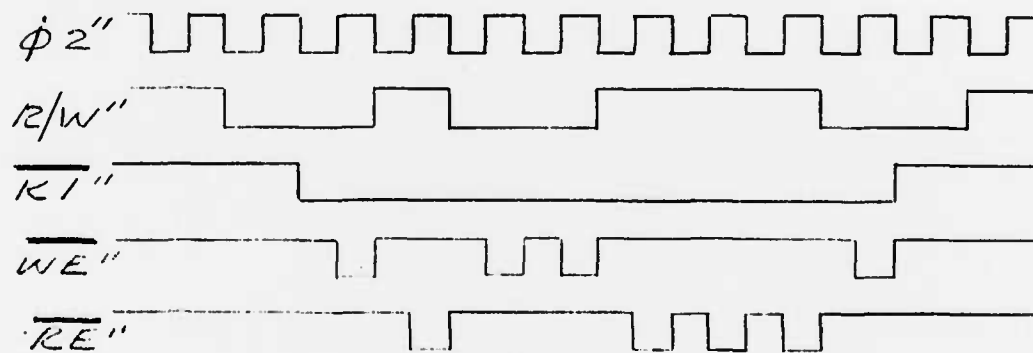
5. When "00" is displayed as a DAC number on the front panel, it does not denote latch of data into any DAC, but rather a mode of operation invoked when latching the Word Data Display (Ref: Paragraph 4.3.2.1).

6. If it is desired that a specified DAC sample a word during every frame, the "Word Interval" must be set to "01".

PROCESSOR 1



PROCESSOR 2



COMMON MEMORY



Figure 3. Common Memory Timing

3.0 HARDWARE

3.1 The heart of the DAC-20 system is the dual 6502B microprocessors which control the data entry, PCM data flow, and the enabling of the DAC's. A block diagram locating circuitry within the front and rear panels (and the five wirewrap cards) is shown in Figure 5, page 12.

3.2 Card 1 - Processors (DWG: D90PE01 and C90PE02)

3.2.1 Processor 1 - IC623 (6502B) controls the latching of the DAC's with 4-to-16 decoders (IC626 and, on Card 2, IC712), and the Binary Word Display (WDE'). IC622 decodes the addresses for the memory and I/O functions. IC625 is 1K of EPROM which contains the processor's operating program.

3.2.2 Processor 2 - IC607 controls data entry from the keyboard and writes the DAC control program for both processors, based on the data entered. IC603 decodes the addresses for the memory and I/O functions. The processor has 2K of EPROM (IC606 and, on card 2, IC708), and 5K of RAM (IC's 601, 604, and, on card 5, IC's 406, 407, 408, 409, 411, 412, 413, 414). The data bus for this processor is buffered on card number 5 for the extra memory, the EPROM, the I/O of card number 2, and the front panel. If the addresses K0", K1", K2", or K7" are selected, then these buffers are disabled by a low on the MEM line. If address lines AB13" and AB15" are not equal, then IC627A disables the IC603 address decoder. The 4K of RAM on card 5 is then enabled by IC404A. Processor 2 monitors the decomp address lines with IC619 and IC620.

3.2.3 CLOCKS - A 9 MHz clock is divided down to 2.25 MHz by IC629. By using the Q output for processor 1 and Q̄ for processor 2, the processors are 180 degrees out of phase. The 6502 microprocessor does its internal functions on phase 1 of the clock and external functions on phase 2.

3.2.4 COMMON MEMORY - Since the processors operate on opposite phases of the clock, they are able to read or write to a common 1K of RAM (IC's 608, 611). Their address lines are multiplexed through IC's 605, 609, and 610. The lines are controlled by one phase of the clock, allowing the memory to be addressed by either processor at any time. When the memory is selected, a read or write to the RAM is determined by IC627D. The timing for the common memory is shown in Figure 3. A common data bus is created by tristate buffers (IC 614 through 619), and is controlled by the Read/Write signals and the appropriate address.

3.2.5 INTERRUPTS - When the DAC processor is powered "ON", a reset interrupt occurs through a one-shot timer (IC602), which resets both processors. The SET and INIT push-switches on the front panel are interrupts and cause processor 2 to execute particular subroutines. The SET and INIT switches also cause processor 1 to reset.

3.3 Card 2 - INTERFACE CARD (DWG: D90PE03 and B90PE04)

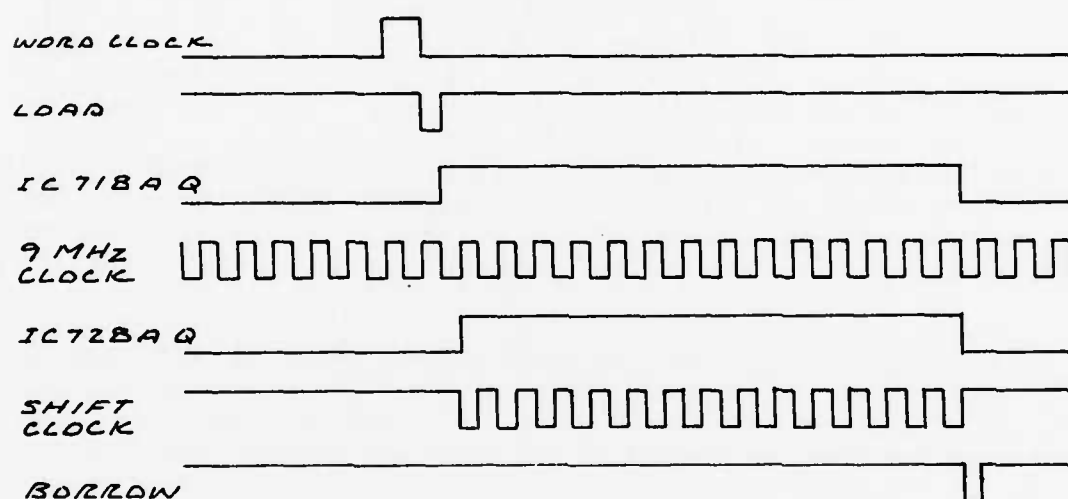


Figure 4. ID Detection Timing

3.3.1 DAC CALIBRATION - During normal operation, the decom data lines are selected by a "high" on the SEL line (pin 12, IC722). Processor 2 puts the specified calibration percentage on the cal latches (IC's 713 through 716) and then sets the SEL to a low to enable the multiplexer (IC's 701 through 704). Processor 1 then enables all DAC latches on cards 3 and 4.

3.3.2 EPROMS - IC708 contains nearly all of the WDACUP routine and its subroutines. (The WDACUP routine is a DAC enable routine for Processor 1, and a frame interval update routine for processor 2.) IC709 is an expansion EPROM. Both EPROMS are processor 2 memory.

3.3.3 ID DETECTION - The delayed word clock is used as timing to the ID detector. On the upward transition of this word clock, the decom data lines are valid and this edge triggers IC718B (JK Flip-Flop). Three inverters of IC727 delay the Q output before being cleared. This provides a 120 nanosecond pulse to load the shift registers with the decom data and enter a preset count into IC723 (4-bit binary counter). This preset count is determined by the location of the ID in the ID word.

The clear pulse of IC718B triggers IC718A, indicating the load pulse is complete. The Q output enables the J and K inputs of flip-flop IC728B. On a positive transition of the 9 MHz clock, Q is high, turning on IC724A. The output of IC724A is the shift register shift clock and the countdown clock for the preset counter. When the counter reaches zero, a "borrow" pulse clears flip-flop IC728A and 718A. The ID is then in the S/P shift register (IC706). The ID detection is summed up in the timing diagram of Figure 5. If the ID is less than eight bits, IC's 725 and 726 turn off the unused bits. Processor 1 then reads either the ID or the decom address lines.

Even though all words are shifted in this circuit, processor 1 looks at the ID only during the ID word.

3.3.4 CURSOR - The cursor position is controlled by processor 2 address lines. By latching these lines (IC730), one of ten cursor positions can be obtained through the IC729 decoder. If address COA" is selected, no decimal point is enabled.

3.3.5 BITS/WORD - IC's 734 and 735 latch in the number of bits/word. The operator has control of turning off any of the twelve least significant bits. IC's 731, 732, and 733 will take low any of the cal/decom data lines not enabled by the latches.

3.3.6 FRONT PANEL INTERFACE - IC719 and 711 enable the write functions on this card and the front panel. IC719B enables processor 2 to read the keyboard data.

3.4 CARD 3 - TEN-BIT DAC CARD (DWG: C90PE05)

3.4.1 DAC'S 1 through 10 are Hybrid Systems model 371-I10 DAC's with current outputs. The incoming data is latched by two 74C174 6-bit

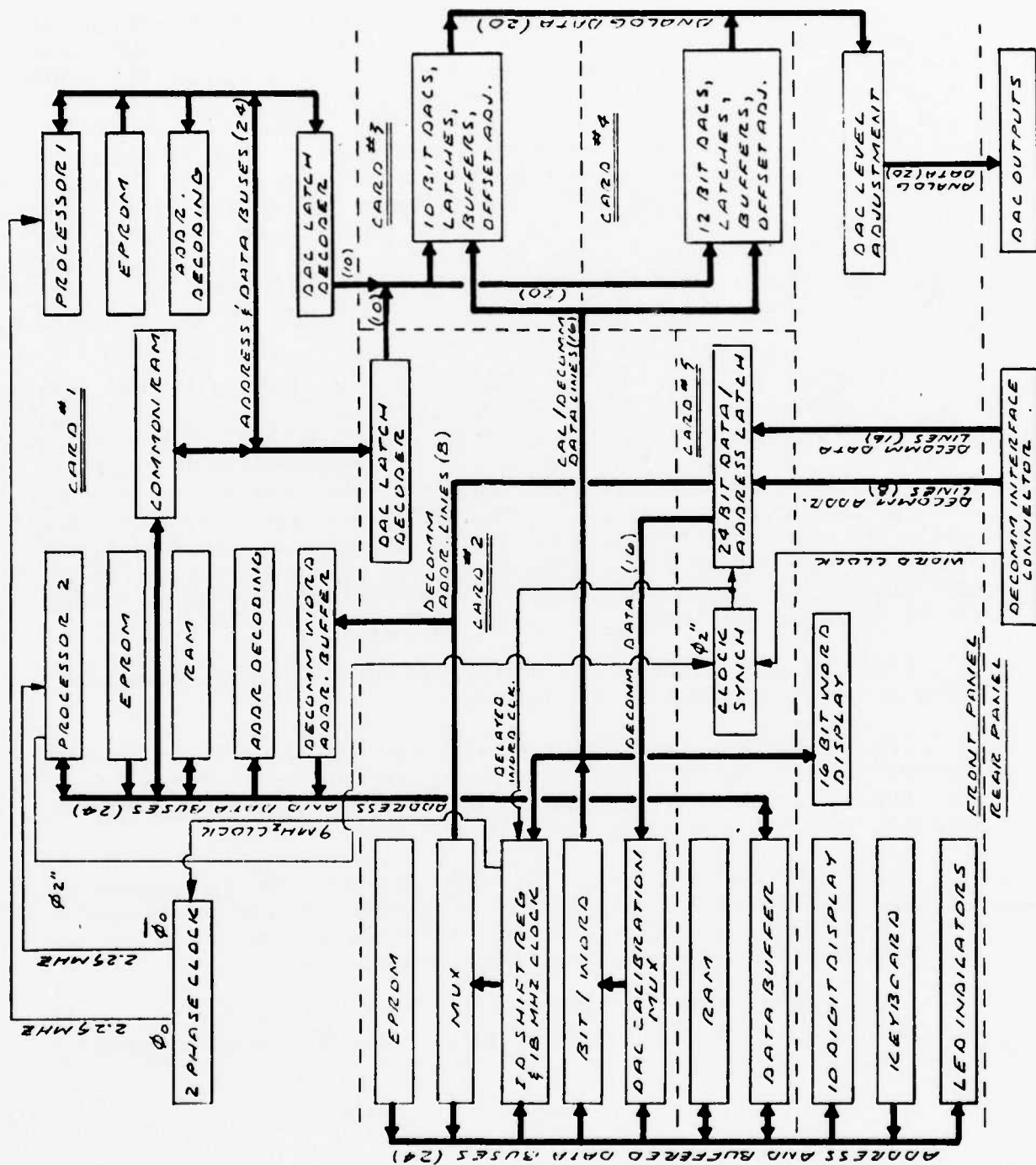


Figure 5. Block Diagram, DAC-20 Hardware

latches. The outputs of the DAC's are converted to a voltage and multiplied by 10 through a 1741S operational amplifier. The output is 0 to +10 volts and a 10K pot is used to adjust the zero offset through approximately ± 0.22 volts.

3.5 CARD 4 - 12-BIT DAC CARD (DWG: C90PE06)

3.5.1 TWELVE BIT DACS - DAC's 11 through 18 are Computer Lab model HDH-1205. These DAC's have -10.24 to 0 volts output. The incoming data is latched by two 74LS174 6-bit latches (IC801 and 802). The output of each DAC is buffered by a 1741S operational amplifier with an output range of 0 to 10.3 volts. The +R (0.4v) is a positive reference, and the -R (-0.6v) is a negative reference for the operational amplifier offset adjustment. The output could be set to a bipolar value if desired.

3.6 CARD 5 - UPDATE RAM CARD (DWG: D90PE07)

3.6.1 This card provides memory space for the UPDATE table for processor 2. IC's 410 and 415 are data bus buffers for the RAM, the interface card and the front panel. IC's 401 and 402 buffer the address lines for the 4K of RAM. For the processors to correctly read the output lines of the decom, these lines must not change during $\emptyset 2$ of either processor. Because of the capacitance of the interface cable, the word clock is delayed 2 microseconds (by IC420). The delayed word clock is then synced with $\emptyset 2$ by IC421. The output, (pin 6 of IC421), is used to enable the four six-bit latches (IC's 416-19) for the decom word and address lines.

3.7 FRONT PANEL (DWG: C90PE08, B90PE09, AND B90PE10)

3.7.1 WORD DISPLAY - A 16-bit binary LED display of the decom data for a selected word is provided. The data is latched at the chosen word time by WDE'.

3.7.2 KEYBOARD ENCODER CARD - Processor 2 scans all of the keys during the INITIALize or SET modes, waiting for a key to be pressed. Two exceptions are the INIT and SET keys, which are interrupts.

Also on this card are the mode indicator LED's. These are latched in by IC's 101 and 102.

3.7.3 TEN DIGIT 7-SEGMENT LED DISPLAY - This unit has 10 individual circuits which latch in the BCD number, to be displayed in decimal form, when written into by processor 2.

3.7.4 DAC LEVEL ADJUSTMENT - A 1K pot is provided for each DAC to adjust the DAC output level.

3.8 REAR PANEL (DWG: B90PE12)

The rear panel is shown in Figure 6.

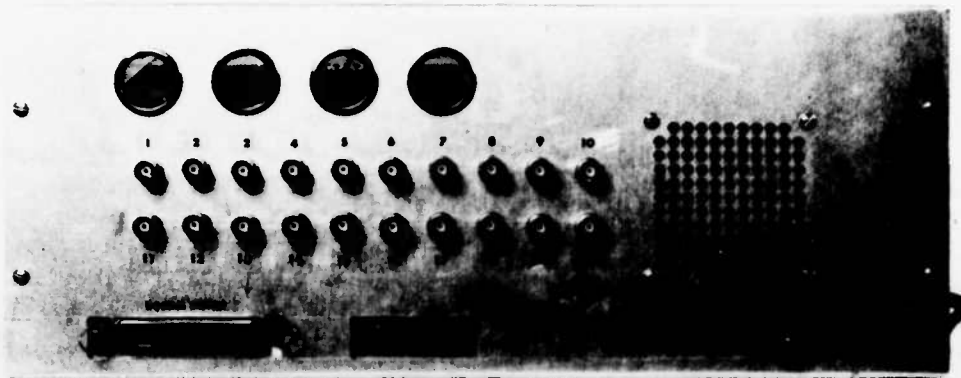


Figure 6. View of Rear Panel, DAC-20

3.8.1 DECOM INPUT - This is the interface line from an OSU decom; any decom which provides the necessary characteristics listed in the table which follows may be used. All inputs are to be TTL levels and positive logic. Signals should be wired to the interface input connector pins as shown in the table which follows:

DATA BITS - Pins 1 through 16; pin 1 is MSB

WORD CLOCK - Pin 18, min 100 nanosec; max 500 nanosec

WORD ADDRESS LINES

BCD UNITS - Pins 21, 22, 24, 28; Bit weight: 1, 2, 4, 8, respectively

BCD TENS - Pins 31, 32, 34, 38; Bit weight: 10, 20, 40, 80 respectively

GROUND - Pins 48 and 50

3.8.2 DAC OUTPUTS - The 20 BNC connectors have 0.1 ufd capacitors to chassis ground. The noise on the output is typically less than 1.5mv. on each output.

4.0 SOFTWARE

4.1 The software for the DAC Processor was assembled and simulated on the Oklahoma State University KIM microcomputer development system. The routines were then transferred in machine code language into the EPROMS.

There are two basic sections of the software. The first provides the data entry and calibration routines, and the second is used in compiling entered data.

4.2 DATA ENTRY AND CALIBRATION - A flow chart describing data entry and calibration is shown in Figure 7.

4.2.1 PROCESSOR 1

4.2.1.1 PROC1. - In this routine, Processor 1 is used to monitor flags from Processor 2. If the calibration flag is enabled, Processor 1 will enable all of the DAC's and the DAC word display. It will then return and wait on another flag from Processor 2. If the RUN flag is enabled, Processor 1 will jump to common memory and execute the DAC Enable routine. This routine is discussed in section 4.3.

4.2.2 PROCESSOR 2

4.2.2.1 POWER. - This routine initializes the INIT workspace. The ID word number is set to 01 and the other locations in the buffer are set to zero. The DAC table is set to zero and the DAC numbers are calculated, then stored from 1 to 32. Other locations in zero page that are used as pointers are initialized.

4.2.2.2 INIT. - This routine is a part of the "power on" reset program. It turns on the "INIT" LED to indicate the mode for the DAC SET display. The INIT buffer is displayed with the ID word number of 01. The cursor is enabled and the routine waits on a data available signal from the keyboard. Once a key has been depressed, the step key is checked; if depressed, the processor jumps to the Bits/Word routine. If not, the routine checks the BCD or BINary keys; if either are indicated, the appropriate LED is enabled to show the ID number system in use. If neither of these keys are depressed, the routine checks for the AUTO or MAN key. If AUTO, a jump to the AUTO subroutine occurs; if the MAN key has been pressed, a jump to the MAN routine is made.

If none of the above keys have been depressed, a check for the cursor keys is made. A cursor right increments the cursor to the right in the DAC SET display. Once in the right most position, the key is ignored. A cursor left decrements the cursor to the left, and when in the left most position, the key is ignored.

If the depressed key is not a control key, then it must be a number. A jump to subroutine NUM occurs and when complete, the initialize buffer is again moved to the display buffer. Then the process of checking for a key begins again.

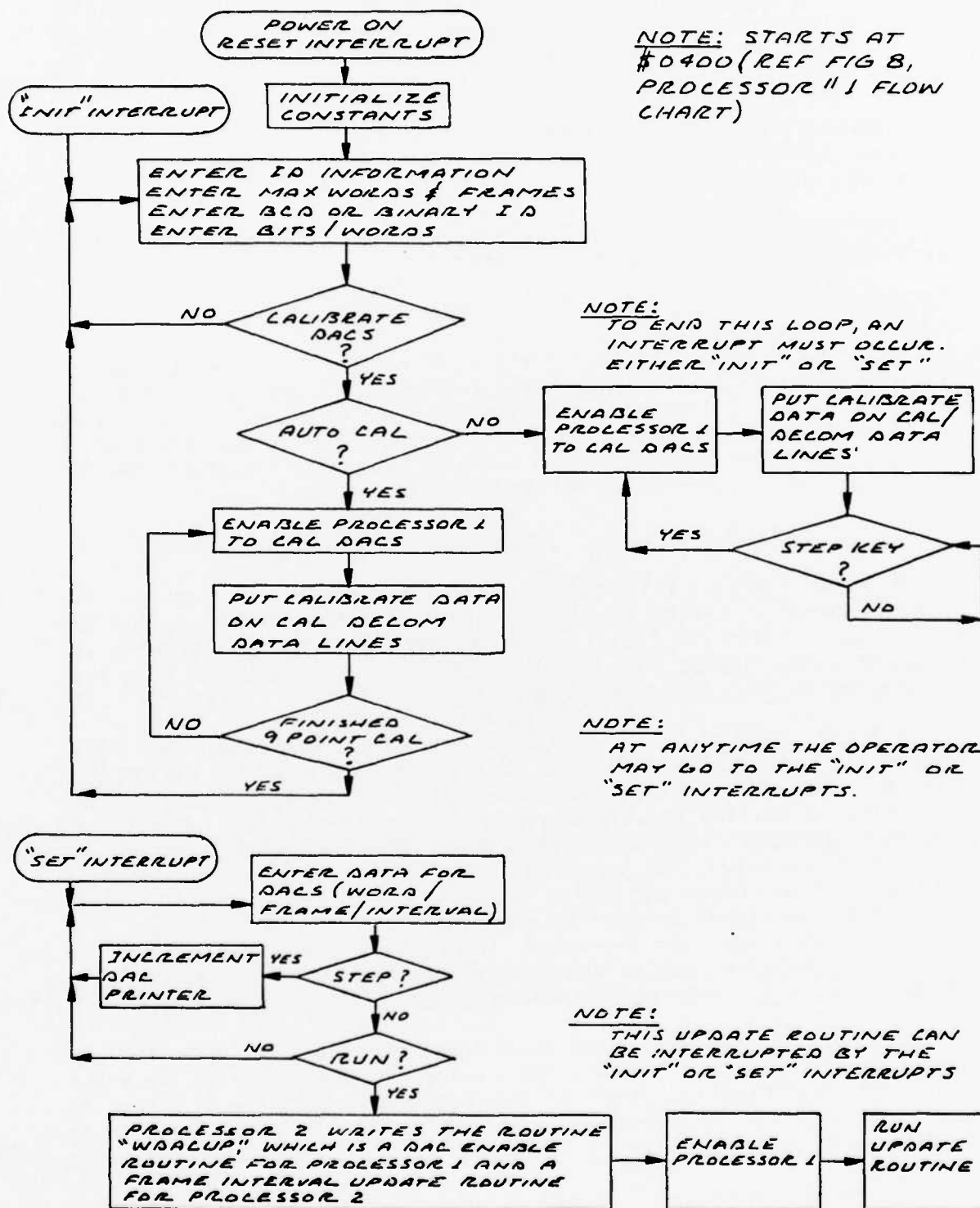


Figure 7. DATA ENTRY Flowchart

4.2.2.3 BITS/WORD - This is a part of the INIT routine and determines the turning on or off of the data bits to the DAC's. The BITS/WORD LED is turned on, along with keeping the INIT and BCD or BIN LED's on. The DAC SET display is initially set to 16, with the right eight digits blanked. Since only the first two digits are used, the cursor may be moved only in these two positions.

If the depressed key is not a cursor key, then the routine checks for a number. The number routine determines the number of Cal/Decom data lines to be blanked. This routine remains in a loop until the INIT or SET interrupts occur.

4.2.2.4 MOVLIN.- This subroutine is used to move a line from the DAC Table (or any 40 Bits from zero page) to the DAC SET display.

4.2.2.5 NUM.- This subroutine is called when a number key has been depressed. The KEYBD subroutine (not used in NUM, but used to decode the keyboard) returns a number in the right four bits in BCD form. The address to store the number is determined by the cursor position, and whether it is in the right or left nibble. The number may be shifted left four bits to set it into the left nibble. The number is combined with predetermined display buffer address data and moved back to the display buffer, and then to either the DAC Table or the "INITialize" buffer.

The cursor pointer is incremented (unless already to the extreme right) and the routine returns to the program that called it.

4.2.2.6 AUTO. - This subroutine turns on the AUTO LED and then calls the CAL routine. When CAL is complete, AUTO returns to the INIT program.

4.2.2.7 MAN. - The MAN routine turns on the MAN LED and then calls the CAL subroutine. MAN continues to call the CAL routine until an interrupt occurs.

4.2.2.8 CAL. - The CAL subroutine saves a pointer indicating an AUTO or MANual mode, and the appropriate LED is turned on. An indicator is set for the direction of an automatic calibration.

A table of constants contains the percentage of calibration blanking for the display, and the sixteen bits of data that actually calibrate the DAC's. The table contains five steps of this calibration data, from 0 percent to 100 percent. A line of this table is moved to the Word display, DAC SET display, and to the Cal latches on card #2. A flag for Processor 1 is enabled, which latches all the DAC's. The AUTO/MAN pointer is checked. If in the MANual mode, the routine waits on the STEP key. If in AUTO, a delay of one second occurs. When either condition is satisfied, a reverse data flag is checked for the AUTO mode. If the flag is set, then the calibration sequence is reversed. When the pointers indicate the end of the data table, the routine ends.

4.2.2.9 SET. - This routine is executed via the SET interrupt. It initializes each selected DAC with a word number, beginning frame number, and a word or frame interval. Initially, the DAC SET LED is

enabled, the display is set to DAC #01, and the cursor points to the word number. The RUN key is checked, and if depressed, a jump to the WDACUP routine occurs. If not, the STEP key is checked and it steps through the DAC Table according to the DAC numbers.

If neither of these keys is depressed, the keyboard is scanned for the cursor left and right keys. These keys move the cursor left or right until the end of the display is reached; then they are ignored. If a key has been depressed and it is none of the above, then it must be a number. The NUM subroutine is called and it puts the number into the display buffer and into the DAC Table.

If the desired DAC number entered is greater than 32, then this DAC number is displayed, but the rest of the DAC SET buffer is blanked. The routine waits on a DAC number less than 32 before unblanking the display. (At present, there are only 18 DAC's in the system. DAC positions 19 through 32 are reserved for future applications.)

4.2.2.10 KEYBD. - The KEYBD subroutine scans the keyboard for a depressed key by columns and rows. Once the column is determined, there is a delay to debounce the key. Based on the column and row, a factor from the table KTAB is added to the column factor Y. This number is used to point to the actual value for the depressed key from the data table KEYTAB. The hexadecimal value for the key is returned in the accumulator.

4.2.2.11 INTVEC. - This is data for the interrupt vectors. The SET interrupt vector is at \$1FFA, the power ON reset at \$1FFC, and the INIT at \$1FFE.

4.2.2.12 DELAY. - Subroutine that delays approximately one second.

4.2.2.13 DEL1. - This routine can delay up to one second, but the delay is determined by the data in the accumulator upon entry to the routine.

4.3 COMPILING THE DATA ENTERED

4.3.1 WDACUP. - This program writes the DAC Enable routine for Processor 1 and the UPDATE routine for Processor 2, based on the DAC table entries. A flowchart for this routine is in Figure 8.

4.3.1.1 Initialization:

DAC table pointer set to current line in DAC SET display.

Flag for MROUT2, insertion set to zero.

Column counter set to zero.

Update table pointer set to address \$3000.

WI flag set to zero.

"STORE AT" for Update routine set to \$0200.

"START" address for Update routine set to \$0200.

Branch to next routine counter set to zero.

"START" for DAC Enable routine set to \$0400.

"GET" data, pointer set to "START" for DAC Enable.

"STORE AT" for DAC Enable begins at \$0400.

Word counter set to 01.
All locations in Update table set to 80.
Word interval stack set to FF.

4.3.2 The data routine DATAID and BEGIN1 are moved to the DAC Enable and Update routine workspaces. The ID word number and BCD or BINARY mode are inserted into the DATAID routine. The maximum frames per major frame is converted from BCD to Binary and saved in zero page.

4.3.2.1 The LSB of the ID determines the preset count for the ID shift circuit. The most significant bits of the ID that are not used are blanked so that the DAC Enable routine will see an eight bit ID of zero. The DAC SET display is checked for a DAC number of zero. (This is not a DAC, but is a mode that enables the DAC Processor to latch the Word DATA display without latching in a DAC.)

4.3.3 SORTING THE DAC TABLE - With the DAC table pointer set to the first line of data, the line is moved to the current line buffer. The word number or the top of the word interval stack is compared to the word counter. If not equal, the DAC table pointer is incremented. The program will continue to scan the table for words equal to the word counter.

Since FLAG was initially set to WI, on the first time through the table will be searched for word intervals. When this search has been completed, FLAG is set to FI and the table pointer is reset to search the table for frame intervals.

After this search, the word counter is incremented and, if it does not equal maximum words, then FLAG is reset to WI and the process is repeated. If the word counter is equal to the maximum number of words in a minor frame, then the ending routines DATEND and UDATED are inserted in the work spaces. Processor 2 then flags Processor 1 to begin enabling the DAC's and waits until Processor 1 is on a major frame sync.

4.3.4 DETECTING A WORD INTERVAL - If Flag points to WI and the word interval is greater than one, then MROUT1 is executed. The word interval stack word number is set to the interval plus the word counter. The address of the DAC line is also stored on the stack. The word interval stack is discussed at the end of this section.

When a word interval is less than or equal to one, the DAC line is ignored and the pointer is incremented to the next line.

4.3.5 DETECTING A FRAME INTERVAL - When FLAG points to FI and the word interval equals zero, the frame interval is compared to 01. If equal, MROUT1 is executed and the word counter is incremented. If not equal, the FL pointer is checked to see if MROUT2 has already been executed for the word number. If it has, the MROUT4 is executed. If not, MROUT2 is executed and FL is set to YES. In either case, a branch distance is calculated and stored in the UPDATE table. (This table is discussed in the next section.) If the frame interval is equal to zero, it bypasses the new frame interval calculation. If the word is to be

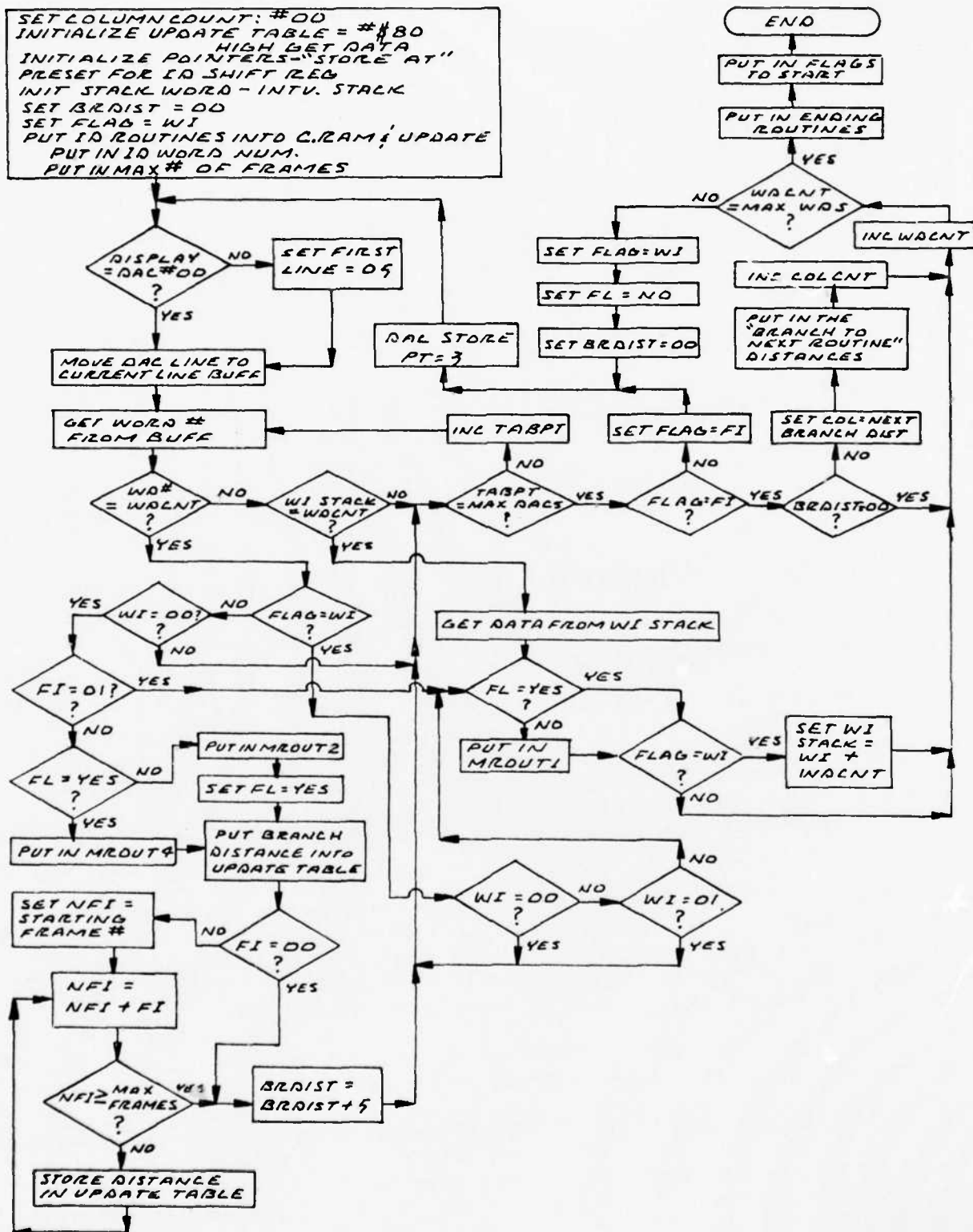


Figure 8. Flowchart For WDACUP

sampled on every frame, the frame interval is set to 01. If the frame interval is greater than one, the new frame interval is set equal to the frame number. New frame intervals are calculated and their branch distances are stored in the UPDATE table until the new frame interval is greater or equal to the maximum number of frames. The branch distance is then set to the branch distance plus five, in case another DAC line has an equal word number but with different frame intervals.

4.3.6 UPDATE TABLE - This table is used by the UDAT2 routine to update the branch distance in DATA2 (which enables a DAC for a selected word with a frame interval). The table is 32 columns by 100 frames and each column represents a different word with a frame interval. A PCM data word may be selected several times with different frame intervals. An example of this is in section 4.3.9.

4.3.7 WORD INTERVAL STACK - Each of the 32 locations in the stack consists of a decomp word address and a DAC Table zero page address. This DAC Table address is the beginning location of a DAC line. The WDACUP routine uses this stack to help sort the DAC Table if a word interval is detected. The tip of the stack is at \$2FFF and the bottom at \$2FC0. When data is pushed into the stack, it is inserted in order of the decomp word address and the lowest word number is at the top of the stack. Data to be read from the stack is pulled from the top and the rest of the stack is pushed to the top.

4.3.7 Data Routines for WDACUP

4.3.7.1 DATAID - This routine is moved to common memory and is used by Processor 1 to find the ID for major frame or minor frame sync. The ID is saved in common memory. This routine is considered to be data until moved.

4.3.7.2 DATA1 - When this data is moved to common memory, processor 1 uses it to find the desired word number (this number and the DAC number are provided by "WDACUP" routine), then enables the DAC latch.

4.3.7.3 DATA2 - This routine is the same as DATA1 except a branch distance is supplied by processor 2. If this branch distance is zero then the DAC is enabled. If not, then it branches past the DAC enable code.

4.3.7.4 DATA4 - This is used with DATA2 to allow multiple DAC's to be enabled on a certain word, but on different minor frames.

4.3.7.5 DATEND - This data, when moved to common memory, allows processor 1 to count the minor frames and compare this with the maximum number of frames. If not the maximum, then it jumps to the minor frame sync in DATAID. If the maximum frame is reached, processor 1 will jump to major frame sync in the DATAID routine. This resets the pointers.

4.3.7.6 BEGIN1 - This routine is data until moved to address 0200. When processor 2 runs the program, it begins at MJSYNC and X is set to the ID. The next time through the loop a comparison is made

until the ID is greater than X. This means there is a new minor frame. If the ID is less than X, then it is a new major frame.

4.3.7.7 UDAT2 - This is data until moved to RAM. When used by processor 2 this routine waits until the decom address lines are greater than the desired word number. This routine then gets a branch distance from the distance table and stores it into the common memory (in the DATA2 routine). The pointer for the table is then incremented.

4.3.7.8 UDATED - This data, when moved to RAM, will be used by processor 2 to reinitialize the distance table pointer. Then it will jump back to BEGIN1.

Subroutines for WDACUP.

4.3.8.1 MBLKUP - Routine that moves a block of data to the update program work space. X is the number of bytes to move and Y is the low byte of the starting location of the routine to be moved. \$E7 and \$E8 are the addresses to "get" the data and \$D9 and \$DA are the "store" locations.

4.3.8.2 MOVE1 - This routine moves 40 bits specified by \$EB and stores it in the display buffer, beginning at \$AA.

4.3.8.3 MROUT1 - This routine moves DATA1 to common memory and puts in the desired word and DAC numbers. Then it jumps to RESPT.

4.3.8.4 MROUT2 - This does the same as MROUT1 except it puts DATA2 into common memory and puts UDAT2 into the update program workspace. This routine also calculates the CRADD address for UDAT2.

4.3.8.5 RESPT - Routine that resets the pointer for the starting locations to insert the data routines into common memory and the update program workspace.

4.3.8.6 DACNUM - This routine converts the DAC number to an address in the DAC table. If a DAC is being monitored by the binary word, display bit 6 for the hexadecimal DAC number will be enabled.

4.3.8.7 BCDBIN - The BCD number in X is converted to hexadecimal and returned in X.

4.3.8.8 DISTAB - This routine calculates an address in the distance table, based on the frame number. The branch distance in \$DD is then stored at the calculated address.

4.3.8.9 MOVBLK - Subroutine to move a block of data from a "GET" location to common memory (the "STORE" location). X is the number of bytes to move and Y is the starting location low byte to move the data.

4.3.8.10 MROUT4 - This routine moves DATA4 to common memory. A desired DAC number is stored into this routine and the branch distance is incremented for the next DAC that may be used.

4.3.9 WDACUP Results

The following tables and programs describe the result of the WDACUP routine. A PCM format, the initialization of the DAC processor, a DAC table, and an update table are given.

In the PCM format example, the letters in the table denote the DAC into which the selected word will be latched. (A-E = DAC Nos. 01-05).

PCM FORMAT

		Frame																			
Frame	Sync.	ID	Word Address																		
Nos.	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	
00	XX	XX	00		A					D					D					D	
01	XX	XX	01		B					D					D					D	
02	XX	XX	02		A					D					D			C		D	
03	XX	XX	03			E				D					D					D	
04	XX	XX	04		A					D					D					D	
05	XX	XX	05		B					D					D					D	
06	XX	XX	06		A					D					D					D	
07	XX	XX	07							D					D					D	
08	XX	XX	08		A					D					D					D	
09	XX	XX	09		B					D					D					D	
10	XX	XX	10		A					D					D					D	
11	XX	XX	11							D					D					D	

Initialization of the DAC-20

- SET: BITS/WORD = 08
- ID WORD NUMBER = 02
- ID MSB = 4
- ID LSB = 8
- MAXIMUM WORDS/MINOR FRAME = 20
- MAXIMUM MINOR FRAMES/MAJOR FRAME = 12
- BCD ID
- DAC number "D" will be monitored on the DAC word display.

DAC TABLE

Program Address	DAC Number	Word Number	Frame Number	Word Interval	Frame Interval
0000	00	00	00	00	00
0005	01	04	00	00	02
000A	02	04	01	00	04
000F	03	17	02	00	12
0014	04	09	00	05	00
0019	05	05	03	00	12
001E	06	00	00	00	00
0023	07	00	00	00	00
0028	08	00	00	00	00
002D	09	00	00	00	00
0032	10	00	00	00	00

A Word Interval stack in the WDACUP routine keeps track of the word numbers for DAC #4. Therefore all that must be known about a selected word on a "word interval" program is the beginning word number and the interval spacing. (Ref: UPDATE Software, page 68.)

For the example given, in which DAC #4 is to be assigned as a "supercomm" output, beginning with word number 9 in each minor frame and sampling at a word interval of 5 (every fifth word thereafter), the UPDATE routine in WDACUP will then generate the following table:

UPDATE TABLE

Program Address	Frame	Column									
		01	02	03	04	05	06	07	08.....	1F	
2000	00	00	05	05	80	80	80	80	80.....	80	
2020	01	05	05	05	80	80	80	80	80.....	80	
2040	02	00	05	00	80	80	80	80	80.....	80	
2060	03	0F	00	05	80	80	80	80	80.....	80	
2080	04	00	05	05	80	80	80	80	80.....	80	
20A0	05	05	05	05	80	80	80	80	80.....	80	
20C0	06	00	05	05	80	80	80	80	80.....	80	
20E0	07	0F	05	05	80	80	80	80	80.....	80	
2100	08	00	05	05	05	80	80	80	80.....	80	
2120	09	05	05	05	80	80	80	80	80.....	80	
2140	10	00	05	05	80	80	80	80	80.....	80	
2160	11	0F	05	05	80	80	80	80	80.....	80	
2180	12	0F	05	05	80	80	80	80	80.....	80	
21A0	13	0F	05	05	80	80	80	80	80.....	80	
'	'	"	"	"	"	"	"	"	".....	80	
'	'	"	"	"	"	"	"	"	".....	80	
'	'	"	"	"	"	"	"	"	".....	80	
'	'	"	"	"	"	"	"	"	".....	80	
2FE0	99	0F	05	05	80	80	80	80	80.....	80	

The two microprocessors within the DAC-20 will then proceed through the routines shown in the following pages to enable the various DAC channels and update the program.

Processor 1 - Routine to enable the DACS

Starting ADDR. 0400

```

400    DATAID    SED            ; Set the decimal mode
401                                LDX #02      ; Look for word 2
403    MAJSYN     CPX $801       ; Find the ID address = Word 2
406                                BNE MAJSYN   ;

408                                LDA $800     ; Get the ID data
40B                                BEQ SAVE     ; IF = 00 then go save ID
40E                                BNE MAJSYN   ; IF ≠ 00 then go find MAJSYN
410    MINSYN     CPX $801

413                                BNE MINSYN   ; Is this minor frame sync
415                                CMP $800     ; Does ID = ID counter?
418                                BNE MAJSYN   ; If not go find major frame sync
41A    SAVE       STA $7FE       ; Save the ID for processor 1

41D    DATA2     LDY #$104      ; Find word #4
41F    LOOP1      CPY $801
422                                BNE LOOP1
424                                BEQ NEXT?    ; Branch to NEXT1 or NEXT2 or No BRANCH

426                                STA $0C00    ; ENABLE DAC #A
429                                BEQ NEXT2    ; IF = Go to next routine
42B    NEXT1      STA #C001      ; Enable DAC #B
42E                                NOP

42F                                NOP
430    NEXT2      LDY #05        ; Find word #5
432    LOOP2      CPY $801
435                                BNE LOOP2

437                                BEQ NEXT?    ; Branch to NEXT3 or No Branch
439                                STA $C004    ; Enable DAC #E
43C    NEXT3      LDY #$09      ; Find word #9
43E    LOOP3      CPY $801

441                                BNE LOOP3
443                                STA $C043    ; Enable DAC #D
                                           ; the 4 in $C043 denotes that
                                           ; the DAC word Data
                                           ; display will be enabled

446                                LDY #$14     ; Find word #14
448    LOOP4      CPY $801
44B                                BNE LOOP4
44D                                STA $C043    ; Enable DAC #D

450                                LDY #$17     ; Find word #17
452    LOOP6      CPY #801
455                                BNE LOOP5
457                                BEQ NEXT?    ; Branch to NEXT4 or No Branch

```

459		STA \$C002	; Enable DAC #C
45C	NEXT4	LDY #\$19	; Find word #19
45E	LOOP6	CPY \$801	
461		BNE LOOP6	
463		STA \$C043	; Enable DAC #D
466	DATEND	CLC	
467		ADC #\$01	; Increment the frame counter
469		CMP \$CC	; Compare with Max frames
46B		BNE MFR	; Check for end of major frame
46D		JMP MAJSYN	; Jump to major frame sync
470	MFS	JMP MINSYN	; Jump to minor frame sync

Processor 2 - Routine to update program at \$0400.

200	BEGIN	CPX \$7FE	; Does X = ID
203		BCC SETX	; IF ID \neq X then go to set X
205		BEQ BEGIN	; IF ID = X then wait
207	MJSYNC	LDY #\$20	; Initialize "Low" table pointer
209		STY \$EA	; Initialize "High" table pointer
20A	SETX	LDX \$7FE	; SET X = ID
20D		LDA *\$04	; Find Word \$4
20F	LOOPA	CMP \$800	
212		BCS LOOPA	; IF word # >decom address lines, then
214		LDA (\$E9),Y	; Get branch distance from update table
216		STA \$425	; and store in processor 1's routine.
219		INY	; Increment pointer for table.
21A		LDA #\$05	; Find word #5
21C	LOOPB	CMP \$800	
21F		BCS LOOPB	; If word # >decom address lines then
221		LDA (\$E9),Y	; Get branch distance from update table
223		STA \$438	; and store in processor 1's routine.
226		INY	; Increment table for pointer
227		LDA #\$17	; Find word #17
229	LOOPC	CMP \$800	
22C		BCS LOOPC	; IF word # >decom address lines then
22E		LDA (\$E9),Y	; get branch distance from update table.
230		STA \$458	; and store in processor 1's routine.
233		INY	; Increment pointer for table
234		TYA	; Get table pointer
235		CLC	
236		ADC #\$1D	; Add # of increments.
238		TYA	
239		BNE KCMJ	
23B		JNC \$EA	; Increment high table pointer
23D		JMP BEGIN	; IF Y = 0.

5.0 OPERATIONAL PROCEDURES-DAC-20

THE DECOM

The OSU Decom must be in "all words" position. Must have parallel data, BCD word address lines, and a positive word clock. Input word rate should not exceed 125 KHz. The DAC processor assumed an incrementing SFID in BCD or binary from 4 to 8 bits wide, located in any bit positions of a word.

THE DAC-20

When power is turned on the unit is reset, and all DAC information is zeroed. The unit is then ready to be initialized (INIT mode).

INIT MODE

ID WORD

The ID may be in any word, but must be at least 4 bits wide. The unit is initialized at word 1. The cursor may be moved left to enter any other ID word number.

ID MSB

Enter the bit location of the MSB of the ID of any given word. (The MSB is the first bit past any fixed bits used as fill in the ID word.)

ID LSB

Enter the bit location of the LSB of the ID. If the ID is less than 4 bits, the MSBs must be zero(s) before the unit will latch in any words. In general, ID cannot be latched into the word data display, since it is used to locate each subframe.

MAX WORDS

This is the total number of words in the minor frame. The minimum number of words is 3 and the maximum is 100 (Enter 00 for 100).

MAX FRAMES

This is the total number of minor frames with a minimum of 1 and a maximum of 100 (Enter 00 for 100).

ID

Enter the mode that is used for the ID: BCD or BIN.

CALIBRATE

The DAC's may be calibrated by percentages of their full scale outputs. These percentages are 0, 25, 50, 75, and 100, and may be manually (MAN key) stepped through (STEP key) or automatically (AUTO key) done under processor control. The unit will remain in the MANUAL mode until the SET or INIT keys are depressed. In the AUTO mode the processor steps through from zero to 100%, back to zero, then returns to the initialize mode.

The DAC outputs on the rear panel have a 0 to +10v range (with approximately 1.5 MV noise), with level adjustments on the front panel, and the zero offsets adjustments on each DAC card. There are ten 10-bit DAC's (DAC's 1-10) with very little offset adjustment, and eight 12-bit DAC's (DAC's 11-18) with zero offset adjustments from +.5v to -6v.

BITS PER WORD

The unit must be in the initialize mode. Depress the STEP key, then enter the number of bits/word with the keyboard. (It is assumed to be 16 bits unless a number is entered.) The processor will turn off the bits not used, both to the DAC's and word display.

SET MODE

This mode is used to initialize the DAC's with the word, frame, word interval, and frame interval information. No two DAC's should have the same information, or the second will be ignored.

The STEP key is used to increment through the DAC's or the cursor may be moved to the DAC position and a number entered to proceed to other DAC information. If the STEP, cursor, or number keys are held down, their function will be repeated.

Once all of the DAC information is entered, then the RUN key may be depressed. This mode sorts the PCM data and latches in the DAC's and WORD display. The DAC information that was last displayed in the SET mode will be latched into the word display.

While in the run mode, if the SET or INIT keys are depressed, the DAC's will not be updated until the RUN key is depressed.

SYSTEM TEST

The easiest test that may be performed to check the operation of each processor is to calibrate the DAC's (either by the manual or auto mode). One processor reads the keyboard and displays the percentage, the other latches all the DAC's and the word display with the binary data. If either function isn't performed, then turn the unit off and back on to reset.

When in the run mode and the DAC's are not being updated, the processors may not be able to obtain the proper subframe ID. Recheck the initialization data to correspond to the desired PCM format.

6.0 OSU PCM DECOM INTERFACE

Pin number (on Cinch 57-40500 Input Connector)

1	MSB
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	LSB
18	Word Clock
21	┐
22	BCD Units
24	Address Lines
28	└
30	+15v
31	┐
32	BCD Tens
34	Address Lines
38	└
40	-15v
41	BCD Hundreds
42	Address Lines
43	Minor Frame Sync
44	Major Frame Sync
45	+5v
48	GND
50	GND

INTENTIONAL

BLANK

PAGE

7.0 PARTS LIST

7.1 Main Frame Details

7.1.1 Power Supply (OSU Dwg. B90PE11) & Chassis

- 1 5v @ 5 amp Supply (Standard Power Supply SPS-40-5)
- 1 ± 15 v Supply (Standard Power Supply SPS-40D-12/15)
- 1 MC7905 -5volt Regulator (Motorola)
- 1 NA7812 +12volt Regulator (National)
- 1 47 ohm $\pm 10\%$ 2watt Resistor (Ohmite)
- 1 Fuseholder (Littlefuse 342012A)
- 1 3AG Fuse, 3 amp Slo-Blow
- 1 AC Power Cord (Belden 17237, 3 wire 18 gauge)
- 1 Blower (Pamotor 8500C)
- 10 70-pin Sockets (Cambion 706-7029-01-00-00)
- 10 Card Guides (Smith 6339)

7.1.2 Rear Panel (OSU Dwg. B90PE12A)

- 20 BNC Connectors Panel Mount (UG-625/U)
- 20 10 Kohm pots (Bourns 273-1-102M)
- 20 0.1uFd 50v Capacitor (CRL CY20C104M)
- 1 50-Pin Decoder Input Connector (Cinch-Jones 57-40500)

7.1.3 Front Panel (OSU Dwg. C90PE08)

- 1 10-digit LED Decimal Display (Dialco 739-1062-601)
- 1 Power Switch (Centralab TV-5)
- 2 Rack Handles, 7" (S/A 26521, Beige)

Keyboard Encoder (OSU Dwg. B90PE09)

- 2 74S175 Quad "D" Flip-Flop (4-bit latch) (National)
- 2 74LS367 Hex 3-State Buffer (National)
- 1 10uFd 35v Capacitor (Mallory MTP106M035PIA)
- 8 150 ohm $\pm 5\%$ $\frac{1}{4}$ watt Resistors (Ohmite)
- 4 1.3 Kohm " " " " (")
- 2 3.3 Kohm " " " " (")
- 2 XC209R LED, 0.125" (National)
- 3 XC556G LED, 0.200" (")
- 5 XC556R LED, 0.200" (")
- 5 16 Pin W.W. Sockets (Garry 102-16-AA-B)
- 1 16 Pin Component Carrier (Cambion 702-3728)
- 4 2-key switch module (Grayhill 82-201-41)
- 2 6-key switch module (" 82-601-81)

Binary Word Display (OSU Dwg. B90PE10)

- 4 74S175 Quad "D" Flip-Flop (4-bit Latch) (National)
- 16 150 ohm $\pm 5\%$ $\frac{1}{4}$ watt (Ohmite)
- 16 XC556R LED, 0.200" (National)
- 8 16-pin W.W. Socket (Garry 102-16-AA-B)
- 4 16-pin Component Carrier (Cambion 702-3728)

7.2 Card #1, Processors (OSU Dwgs. D90PE01B, C90PE02B)

- 1 74LS00 Quad 2-Input NAND Gate (T.I.)
- 2 74LS04 Hex Inverter (Motorola)
- 2 74LS10 Triple 3-Input NAND Gate (I.T.T.)
- 2 74LS42 4-line BCD to Dec. Decoder (Fairchild)
- 1 74LS86 Quad 2-Input ExOR Gate (Motorola)
- 1 74LS109 Dual "JK" Flip-Flop (National)
- 1 74C154 4-line Binary to 16 Decoder (National)
- 3 74LS157 Quad Data Selector Switch (National)
- 7 74LS367 3-State Hex Buffer (T.I.)
- 2 2114A-L4 1024 X 4-bit RAM (Intel)
- 2 2148 1024 X 4-bit RAM, 90 nanosec (Intel)
- 2 TMS2708TL 1024 X 8-bit EPROM, 250 nanosec (Intel)
- 2 SYP6502B 8-bit Microprocessor (Synertec)
- 1 LM556C Dual Timer (National)
- 11 0.1ufd 50v Capacitor (CRL CY20C104M)
- 3 0.22ufd 50v " (CRL CW20C224M)
- 3 10ufd 35v " (Mallory MTP106M035PIA)
- 1 15ufd 20v " (" MTP156M020PIA)
- 1 1.0K \pm 5% $\frac{1}{4}$ watt Resistor (Ohmite)
- 1 3.0K " " "
- 7 3.3K " " "
- 1 10K " " "
- 2 47K " " "
- 2 40-pin W.W. Sockets (Augat 540-AG-10F)
- 3 24-pin " " (" 524-AG-10F)
- 2 18-pin " " (" 518-AG-10F)
- 11 16-pin " " (Garry 102-16-AA-B)
- 13 14-pin " " (" 102-14-AA-B)
- 5 8-pin " " (" 102-08-AA-B)
- 5 8-pin Component Carriers (Cambion 702-3720)
- 1 Dual 70-pin W.W. Card (Cambion 714-1015-01)

7.3 Card #2, Decom Interface (OSU Dwg. D90PE03B & C90PE04A)

- 1 74LS00 Quad 2-Input NAND Gate (Fairchild)
- 1 74LS04 Hex Inverter (Motorola)
- 3 74LS08 2-Input AND Gate (National)
- 1 74LS10 Triple 3-Input NAND Gate (I.T.T.)
- 1 74LS42 4-line BCD to Dec. Decoder (National)
- 2 74LS109 Dual "JK" Flip-Flop (National)
- 1 74LS126 Quad 3-State Buffer (National)
- 2 74154 4-line Binary to 16 Decoder (National)
- 4 74LS157 Quad 2-Input Data Selector Switch (National)
- 1 74LS164 8-Bit Ser. In/Par.Out Shift Register (T.I.)
- 2 74LS165 8-Bit Par. In/Ser. Out " " (Fairchild)
- 4 74LS174 Hex "D" Flip-Flop (6-bit latch)(T.I.)
- 5 74LS175 Quad "D" Flip-Flop (4-bit latch)(National)
- 1 74LS193 Presettable 4-bit Binary Up/Down Counter (T.I.)
- 2 74LS257 Quad 3-State Data Selector (Motorola)
- 1 74C901 Hex Inverter (National)
- 1 TMS2708-1 1024 X 8-bit EPROM, 250 Nanosec (Intel)
- 10 0.1ufd 50v Capacitor (CRL CY20C104M)
- 3 10ufd 35v " (Mallory MTP106M035PIA)
- 1 18.00 MHz, Crystal, CY19A
- 4 1.1K \pm 5% $\frac{1}{4}$ watt Resistor (Ohmite)
- 1 150K " " " " (")
- 4 24-pin W.W. Socket (Augat 524-AG-10F)
- 22 16-pin " " (Garry 102-16-AA-B)
- 11 14-pin " " (" 102-14-AA-B)
- 2 14-pin Component Carrier (Cambion 702-3725)
- 1 Dual 70-pin W.W. Card (Cambion 714-1015-01)

7.4 Card #3, 10-Bit DAC Card (OSU Dwg. C90PE05)

- 20 74C174 Hex "D" Flip-Flop (6-bit Latch) (National)
- 10 3711-10 10-Bit DAC (Hybrid Systems)
- 10 MC1731SCP1 Operational Amplifier (Motorola)
- 3 10ufd 35v Capacitor (Mallory MTP106M035PIA)
- 10 270 ohm \pm 5% $\frac{1}{4}$ watt Resistors (Ohmite)
- 10 1.0Kohm " " " (")
- 10 10Kohm " " " (")
- 10 10K Pots (Bourns 3282H-1-103)
- 30 16-pin W.W. Sockets (Garry 102-16-AA-B)
- 5 14-pin " " (" 102-14-AA-B)
- 15 8-pin " " (" 102-08-AA-B)
- 10 8-pin Component Carriers (Cambion 702-3720)
- 1 Dual 70-pin W.W. Card (Cambion 714-1015-01)

7.5 Card #4, 12-Bit DAC Card (OSU Dwg. C90PE06)

- 16 74LS174 Hex "D" Flip-Flop (6-bit Latch) (T.I.)
- 8 HDH-1205 12-Bit DAC (Computer Labs)
- 8 MC1741SCP1 Operational Amplifier (Motorola)
- 1 1N5059 200PIV Diode (G.E.)
- 3 10ufd 35v Capacitor (Mallory MTP106M035PIA)
- 1 22ufd 50v " (" MTP226M050PIA)
- 8 10Kohm Pots (Bourns 3282H-1-103)
- 8 24-Pin W.W. Sockets (Augat 524-AG-10F)
- 16 16-Pin " " (Garry 102-16-AA-B)
- 4 14-Pin " " (" 102-14-AA-B)
- 8 8-Pin " " (" 102-08-AA-B)
- 4 8-Pin Component Carrier (Cambion 702-3270)
- 1 Dual 70-Pin W.W. Card (Cambion 714-1015-01)

7.6 Card #5, Update RAM (OSU Dwg. D90PE07A)

- 1 74LS00 Quad 2-Input NAND Gate (T.I.)
- 1 74LS04 Hex Inverter (T.I.)
- 1 74LS42 4-Line BCD to Dec Decoder (T.I.)
- 1 74LS109 Dual "JK" Flip-Flop (National)
- 4 74LS174 Hex "D" Flip-Flop (6-bit Latch) (T.I.)
- 2 74LS367 Hex 3-State Buffer (T.I.)
- 1 DM8602J Dual One-Shot MV (National)
- 2 DS8833 Quad 3-State Bus Transceiver (National)
- 8 2114AL-4 1024 X 4-bit RAM (Intel)
- 1 750 pfd 200v Capacitor (ARCO DM751K)
- 3 0.1ufd 50v " (CRL CY20C104M)
- 1 10ufd 35v " (Mallory MTP106M035PIA)
- 1 10Kohm $\pm 5\%$ $\frac{1}{4}$ watt Resistor (Ohmite)
- 8 18-Pin W.W. Sockets (Augat 518-AG-10F)
- 11 16-Pin " " (Garry 102-16-AA-B)
- 2 14-Pin " " (" 102-14-AA-B)
- 1 Dual 70-pin W.W. Card (Cambion 714-1015-01)

APPENDIX

Processor-Controlled DAC-20 (Software and Flow Charts)

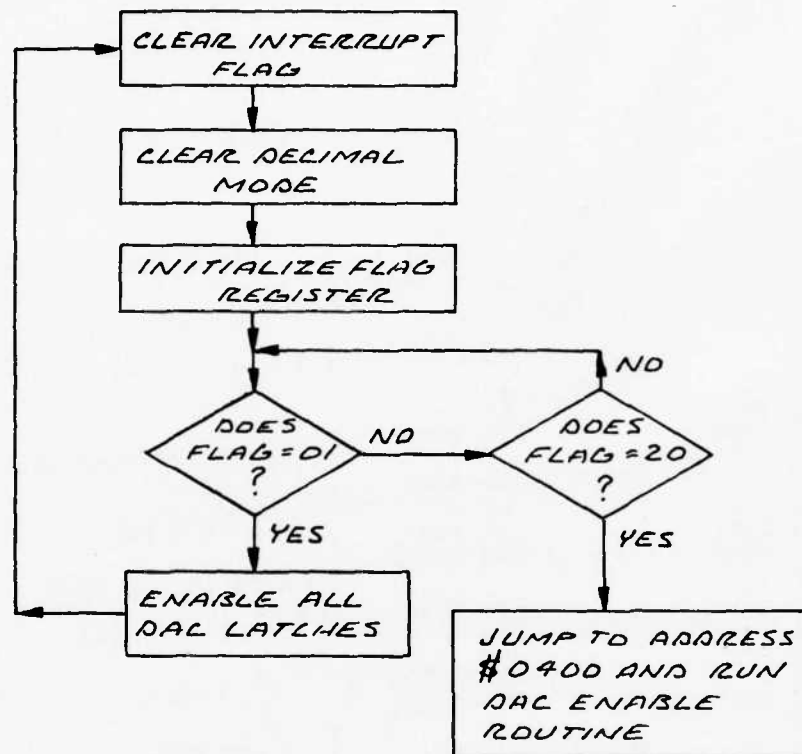
Topic	Page
Processor No. 1 - Memory Map.....	36
Flow Chart.....	37
PROC 1 Routine Software.....	38
Processor No. 2 - Memory Map.....	39
Zero Page.....	42
SET Routine - Flow Chart.....	43
Software.....	45
POWER ROUTINE - Flow Chart.....	47
Software.....	49
NUM Subroutine - Flow Chart.....	52
Software.....	53
DISPX - Data Tables for INIT - BITS/WORD.....	53
CALIBRATION - Flow Chart.....	54
AMCAL Subroutine - Software.....	56
KEYBOARD Subroutine - Flow Chart.....	58
KEYBD - Software.....	59
WDACUP Routine Software.....	60
WDACA (Part A).....	60
WDACB (Part B).....	64
Subroutines for WDACUP.....	67
SUBRT - Data for Processor 1.....	67
UPDATE - Data for Processor 2.....	68

PROCESSOR 1
MEMORY MAP 6/16/81

ADDRESS	ENABLE	USE
0000 - 03FF	$\overline{K0}$	1K RAM (CARD #1) (IC601 & IC604)
0400 - 07FF	$\overline{K1}$	1K COMMON RAM (CARD # 1) (IC608 & IC611)
0800	$\overline{K2}$	ID (CARD # 2) (IC720 & IC721)
0801	$\overline{K2}$	DECOM ADDRESS LINES (CARD # 2) (IC 726)
0C00 - 0C0F	$\overline{AK3}$	ENABLE DACS 1 THROUGH 16 (CARD 3 & 4)
0C10 - 0C1F	$\overline{AK3}$	ENABLE DACS 17 THROUGH 32 (CARD 4)
0CXX - 0CXX	\overline{WDE}	IF ADDRESS BIT 6 IS HIGH THEN THE 16 BIT DISPLAY IS ENABLED (FRONT PANEL)
1C00 - 1FFF	$\overline{K7}$	EPROM - CONTAINS PROC 1 MONITOR (CARD # 1) (IC625)

ADDITIONAL NOTES:

07FE	$\overline{K1}$	ID FROM PROC 2
07FF	$\overline{K1}$	FLAG REGISTER: 00 - WAIT 01 - CALIBRATE 02 - EXECUTE DAC ENABLE PROGRAM



NOTE: WHEN RUNNING THE DAC ENABLE ROUTINE, TO RETURN TO THE FIRST OF THIS ROUTINE THE "SET" INTERRUPT MUST BE ENABLED.

Figure 9. Processor 1 Flowchart

A

```

10 0000      ; PROC1
20 0000      ; 5/7/80
30 0000      ; THIS PROCESSOR DOESN'T USE A STACK OR Z-PAGE
40 0000      ; PROC1 WAITS ON FLAGS (@$7FF) FROM PROC2
60 1C00      *=$1C00
70 1C00
80 1C00      DACS   = $0C40      ; THE 4 ENABLES THE WORD DISPLAY
90 1C00 58    PROC1  CLI
100 1C01 D8    CLD
110 1C02 A900   LDA  #$00
120 1C04 8DFF07 STA  $7FF      ; INITIALIZE FLAG REG.
130 1C07 ADFF07 START  LDA  $7FF ; DOES REG=01
140 1C0A 58     CLI
150 1C0B C901   CMP  #$01      ; THIS IS A CAL
160 1C0D D00A   BNE  FLAG2
170 1C0F        ; LATCH IN ALL DACS
180 1C0F A21F   LDX  #$1F      ; STEP THRU THE DACS
190 1C11 9D400C ENDAC STA  DACS,X ; ENABLE THE DACS
200 1C14 CA     DEX
210 1C15 10FA   BPL  ENDAC
220 1C17 30E7   BMI  PROC1
230 1C19
240 1C19 C920   FLAG2 CMP  #$20 ; CK FOR RUN
250 1C1B D0EA   BNE  START
260 1C1D 4C0004 JMP  $400
270 1C20
280 1C20      ; INTERRUPT VECTORS
290 1FFA      *=$1FFA
300 1FFA 00    NMI    .BYTE  $00,$1C
310 1FFB 1C
320 1FFC 00    RST    .BYTE  $00,$1C
330 1FFD 1C
340 1FFE 00    IRQ    .BYTE  $00,$1C
350 1FFF 1C

```

PROCESSOR # 2
MEMORY MAP

ADDRESS	ENABLE	DESCRIPTION
0000 - 03FF	$\overline{K0}$ "	1K RAM----- $\overline{K0}$ "
0400 - 07FF	$\overline{K1}$ "	1K COMMON RAM----- $\overline{K1}$ "
0800	$\overline{K2}$ "	DECOM ADDRESS LINE----- $\overline{K2}$ "
0C00	$\overline{K3}$ " - C00"	ENABLE DECIMAL POINTS DAC TENS-----
0C01	$\overline{K3}$ " - C01"	" " " DAC UNITS
0C02	$\overline{K3}$ " - C02"	" " " WORD # TENS
0C03	$\overline{K3}$ " - C03"	" " " WORD # UNITS
0C04	$\overline{K3}$ " - C04"	" " " FRAME # TENS
0C05	$\overline{K3}$ " - C05"	" " " FRAME # UNITS
0C06	$\overline{K3}$ " - C06"	" " " WORD INTV TENS
0C07	$\overline{K3}$ " - C07"	" " " WORD INTV UNITS
0C08	$\overline{K3}$ " - C08"	" " " FRAME INTV TENS
0C09	$\overline{K3}$ " - C09"	" " " FRAME INTV UNITS
0C10	$\overline{K3}$ " - C10	LATCH LED PAIRS DAC UNITS, TENS
0C11	$\overline{K3}$ " - C11	" " " WORD #UNITS, TENS
0C12	$\overline{K3}$ " - C12	" " " FRAME # UNITS, TENS
0C13	$\overline{K3}$ " - C13	" " " WORD INTV UNITS, TENS
0C14	$\overline{K3}$ " - C14	" " " FRAME INTV UNITS, TENS
0C15	$\overline{K3}$ " - C15	LATCH CALIBRATION DATA (BITS 1 - 8)
0C16	$\overline{K3}$ " - C16	" " " (BITS 9 - 16)
0C17	$\overline{K3}$ " - C17	LATCH LED INDICATORS - BIT PATTERN
		INDICATORS 7 6 5 4 3 2 1 0
		BCD 0 0 0 0 0 0 0 1
		BIN 0 0 0 0 0 0 1 0
		DAC SET 0 0 0 0 0 1 0 0
		INIT 0 0 0 0 1 0 0 0
		MAN 0 0 0 1 0 0 0 0
		AUTO 0 0 1 0 0 0 0 0
		RUN 0 1 0 0 0 0 0 0
		BITS/WORD 1 0 0 0 0 0 0 0
0C18	$\overline{K3}$ " - C18	LATCH ID COUNTER BITS 0 - 3 ARE ID COUNT
0C19		SELECT DECOM ADDR LINES BIT 4 = SEL CAL OR DECOM
0C1A		NOT USED
0C1B		SPARE
0C1C		TURNS OFF UNUSED BITS OF ID
0C1D		BITS/WORD LATCH (BITS 11 - 16)
0C1E		BITS/WORD LATCH (BITS 5 - 10)
0C1E-0C1F		NOT USED-----

$\overline{K3}$ "

PROCESSOR # 2
MEMORY MAP (CONT'D)

ADDRESS	ENABLE	DESCRIPTION	
1001	$\overline{K4}''$	KEYBOARD COL 1-----	$\overline{K4}''$
1002		COL 2	
1004		COL 3	
1008		COL 4	
		ROWS OF THE KEYBOARD ARE DETERMINED BY THE VALUE OF THE DATA BUS (BITS 0-3)	
		COL 1 2 3 4	
		ROW 1 0 1 2 3	
		2 4 5 6 7	
		3 8 9 <- ->	
		4 man auto bcd bin	
1010		RUN/STEP RUN: BIT 7 = 0; STEP: BIT 5 = 0-----	
EPROM ALGORITHMS ADDRESS 1800-1FFF			
1800 - 1867	$\overline{K6}''$	DATAID CONTAINS :-----	$\overline{K6}''$
186D - 18AE		DATA1	
		DATA2	
		DATA4	
		DATEND	
		UDATID	
		UDAT2	
		UDATED	
		MBLKUP	
18AF - 1889		MOVEL	
188A - 1913		MROUT1 CONTAINS:	$\overline{K7}''$
		MROUT2	
		RESPT	
1914 - 1923		DACNUM	
1924 - 1943		BCDBIN	
1944 - 1960		DISTAB	
1961 - 1974		MOVBLK	
1975 - 1989		MROUT4	
1868 - 186C		NOP'S-----	
1A00 - 1C75	$\overline{K6}''/\overline{K7}''$	WDACUP-----	
1C80 - 1D2A	$\overline{K7}''$	SET	$\overline{K7}''$
1D40 - 1D71		POWER	$\overline{K7}''$
1D72 - 1E92		INIT	

(Cont'd)

PROCESSOR # 2
MEMORY MAP (CONT'D)

ADDRESS	ENABLE	DESCRIPTION
1EC9 - 1EF2	$\overline{K7}$ "	NUM-----
1EF3 - 1EFF		MOVLIN
1F00 - 1F07		AUTO AMCAL
1F08 - 1F11		MAN
1F12 - 1F63		CAL
1F69 - 1F77		DELAY
1F6B - 1F77		DEL1
1F78 - 1F84		DEL
1F85 - 1F9D		REVRS
1FA9 - 1FF9		KEYBD
TABLES:		
1E95 - 1EB7		DATA
1EB8 - 1EBB		JAK
1EBC - 1EBF		DISPX
1ECO - 1EC6		BWTB
1FE1 - 1FE9		KTAB
1FEA - 1FF9		KEYTAB
1FFA - 1FFF		INTVEC-----

$\overline{K7}$ "

PROCESSOR "2" ZERO PAGE

DAC TABLE STORAGE						ADD	DESCRIPTION	SUBROUTINE
ADD	DAC	WRD	FRM	WRD	FRM			
LOC	NUM	NUM	NUM	INT	INT			
00	00	--	--	--	--	BC	USED FOR NIBBLE "OR"ING	INIT
05	01	--	--	--	--	BD	DISPLAY BUFFER FOR BW	INIT
0A	02	--	--	--	--	BE		
0F	03	--	--	--	--	BF	SAVE X POINTER	INIT
14	04	--	--	--	--	C0	SYNC	UPDATED
19	05	--	--	--	--	C1	SEARCH	UPDATED
1E	06	--	--	--	--	C2	SAVE X	SET
23	07	--	--	--	--	C3	SAVE ACCUM	SET
28	08	--	--	--	--	C4	WORD INTV STACK POINT LOW	WDACUP
2D	09	--	--	--	--	C5	WORD INTV STACK POINT HIGH	WDACUP
32	10	--	--	--	--	C6	SAVE Y	SET, MROUT4
37	11	--	--	--	--	C7	MCAL PROGRAM	
3C	12	--	--	--	--	C8	DELAY WORKSPACE	DEL
41	13	--	--	--	--	C9	SAVE TAB PNTR LOW BYTE - NUM: FF	CAL
46	14	--	--	--	--	CA	SAVE NUM WORKSPACE - NUM: FE	CAL, DELAY
4B	15	--	--	--	--	CB	FL - FLAG INDICATE RT #2 INSERTED	WDACUP
50	16	--	--	--	--	CC	SAVE MAX FR (BCD OR BIN)	DACUP
55	17	--	--	--	--	CD	IND - STATUS INDICATOR BUFFER	INIT, SET
5A	18	--	--	--	--	CE	# BYTES TO MOVE	MOVBLK
5F	19	--	--	--	--	CF	WDCNT - WORD COUNTER	WDACUP
64	20	--	--	--	--	D0	UPDATE TABLE POINTER - LOW	DISTAB
69	21	--	--	--	--	D1	UPDATE TABLE POINTER - HIGH	DISTAB
6E	22	--	--	--	--	D2	COLUMN COUNTER	DISTAB
73	23	--	--	--	--	D3	ADDN - WORKSPACE	BCDBIN
78	24	--	--	--	--	D4	BIN - WORKSPACE	BCDBIN
7D	25	--	--	--	--	D5	NFI	WDACUP
82	26	--	--	--	--	D6	UPDATE TABLE ADDRESS - LOW	WDACUP
87	27	--	--	--	--	D7	UPDATE TABLE ADDRESS - HIGH	WDACUP
8C	28	--	--	--	--	D8	FLAG FOR WI = 00, FI = 80	WDACUP
91	29	--	--	--	--	D9	LOW "STORE AT" FOR UPDATE	MBLKUP
96	30	--	--	--	--	DA	HIGH "STORE AT" FOR UPDATE	MBLKUP
9B	31	--	--	--	--	DB	STARTING LOC FOR "STORE AT" - LOW	WDACUP
A0	32	--	--	--	--	DC	STARTING LOC FOR "STORE AT" - HIGH	WDACUP
A5	--	FF	FF	FF	FF	DD	"BRANCH TO NEXT ROUTINE" COUNTER	
						DE		
						DF		
						E0	WORD # OF ID.....	
						E1	MSB OF ID.....	
						E2	LSB OF ID.....	- INIT BUFFER
						E3	MAX # OF WORDS/FRAME..	
						E4	MAX # OF FRAMES.....	
						E5	STARTING LOC LOW OF "STORE AT"	WDACUP
						E6	STARTING LOC HIGH OF "STORE AT"	WDACUP
						E7	LOC LOW TO "GET" (INSERTED DATA)	WDACUP
						E8	LOC HIGH TO "GET" (INSERTED DATA)	WDACUP
						E9	LOW "STORE AT" FOR DAC ENABLE	WDACUP
						EA	HIGH "STORE AT" FOR DAC ENABLE	WDACUP
						EB	DAC TABLE LOW POINTER	
						EC	DAC TABLE HIGH POINTER	WDACUP
						ED	CURRENT LINE IN DISPLAY (LOW)	WDACUP

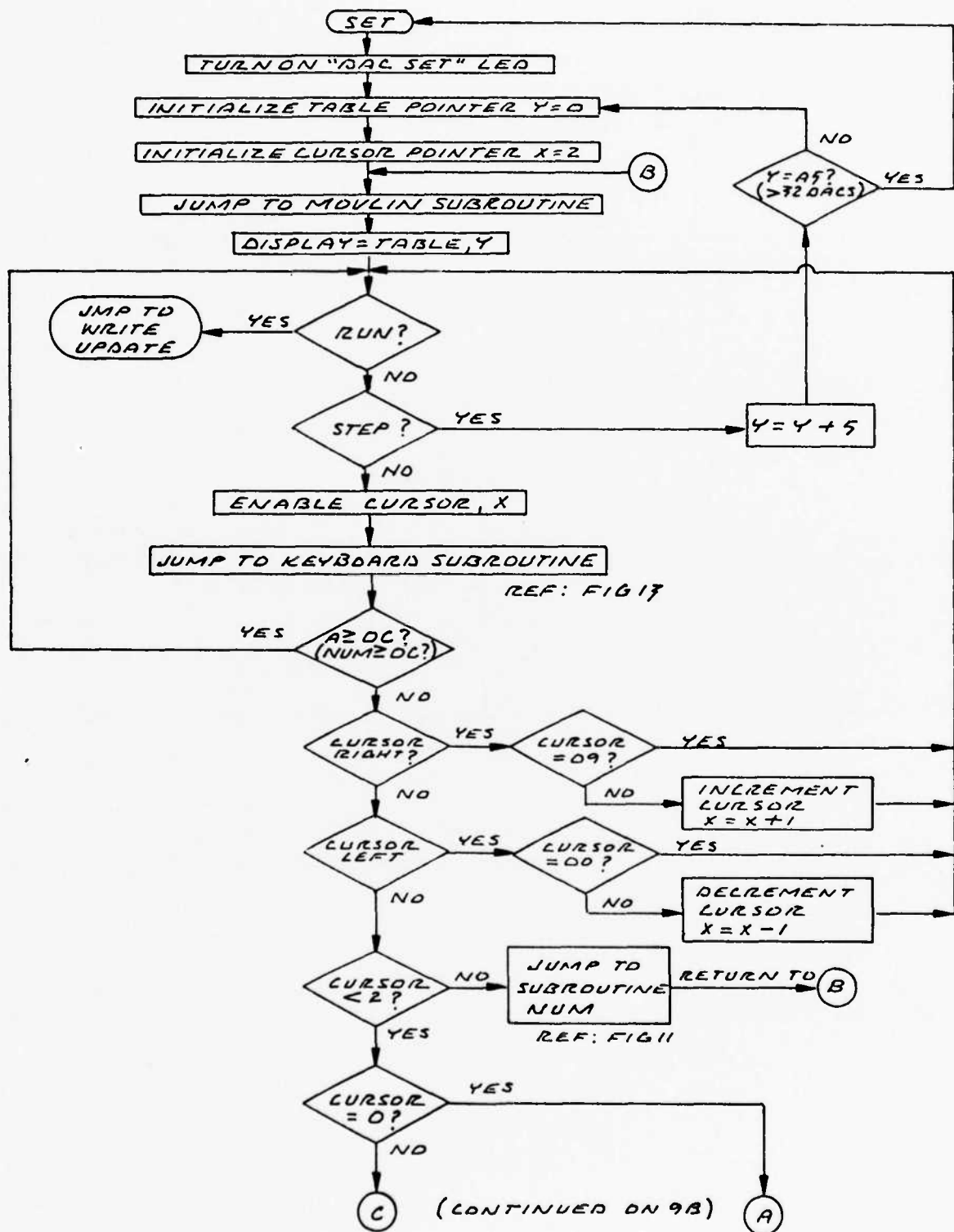


Figure 10A. SET Flowchart

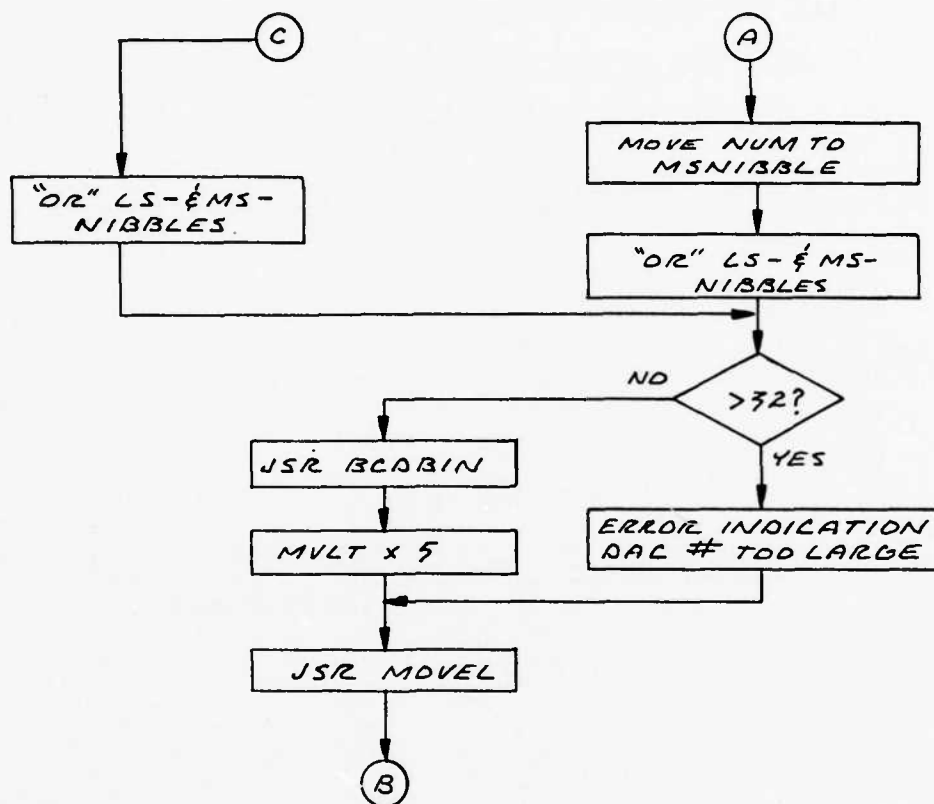


Figure 10B. SET Flowchart (Cont'd)

Software for SET Routine (Start)

A

```

10 0000      ; SET
20 0000      ; 2/22/83
30 0000      ; ROUTINE TO SET THE VALUES IN THE DAC TABLE
40 0000      ; VIA THE KEYBOARD..
50 0000      KEYBD = $1FA9
60 0000      NUM  = $1EC9
70 0000      MOVLIN = $1EF3
80 0000      IND   = $00CD
90 0000      RUNSTP = $1010
100 0000     DEL1  = $1F6B
110 0000     WDACUP = $1A00
120 0000     BCDBIN = $1924
130 0000     MOVEL  = $1BAF
140 1CB0     * = $1CB0
150 1CB0     ;
160 1CB0 A2FF SET      LDX  #$FF
170 1CB2 9A      TXS      ; REINITIALIZE STACK
180 1CB3 86A6     STX  $A6
190 1CB5 86A7     STX  $A7
200 1CB7 86A8     STX  $A8
210 1CB9 86A9     STX  $A9
220 1CB8 A5CD     LDA  IND
230 1CB8 2903     AND  #$03
240 1CBF 0904     ORA  #$04
250 1C91 8D170C   STA  $0C17 ; TURN ON SET LED WITH BCD/BIN
260 1C94         ;
270 1C94 DB      CLD
280 1C95 5B      CLI      ; CLEAR INT VECTOR FOR INIT
290 1C96 A005     LDY  #$05 ; INIT DAC TABLE POINTER
300 1C98 A202     LDX  #$02 ; INIT CURSOR POINTER
310 1C9A 20F31E   MLTAB JSR  MOVLIN ; MOVE A LINE FROM DAC
320 1C9D         ; TABLE TO THE DISPLAY
330 1C9D A000     LDY  #$00
340 1C9F B1EB     LDA  ($EB),Y
350 1CA1 85AA     STA  $AA
360 1CA3 A4EB     LDY  $EB      ; RESTORE DAC TABLE POINTER
370 1CA5         ;
380 1CA5 AD1010   CKKEY LDA  RUNSTP ; TEST FOR RUN OR STEP
390 1CA8 3003     BMI  CKSTEP
400 1CAA 4C001A   JMP  WDACUP      ; GO WRITE THE MAIN PROG
410 1CAD 0A      CKSTEP ASL  A
420 1CAE 0A      ASL  A      ; CK STEP KEY
430 1CAF 0A      ASL  A
440 1CB0 B011     BCS  CKDATA
450 1CB2 A958     LDA  #$58
460 1CB4 206B1F   JSR  DEL1      ; DEBOUNCE THE KEY
470 1CB7 1B      CLC      ; INC Y BY 5
480 1CB8 DB      CLD
490 1CB9 9B      TYA
500 1CBA 6905     ADC  #$05
510 1CBC C9A5     CMP  #$A5      ; BEYOND 32ND DAC?
520 1CBE B0C0     BCS  SET
530 1CC0 AB      TAY
540 1CC1 D0D5     BNE  CURSOR
550 1CC3         ;
560 1CC3 B4C6     CKDATA STY  $C6
570 1CC5 BC000C   LDY  $C00,X ; ENABLE DEC PT
580 1CC8 9A      TXS
590 1CC9 20A91F   JBR  KEYBD      ; GET DATA FROM KEYBOARD
600 1CCC A4C6     LDY  $C6
610 1CCE BA      TBX
620 1CCF C90C     CMP  #$0C      ; IF KEY >= 0C THEN WRONG
630 1CD1 B0D2     BCS  CKKEY

```

Software for SET Routine (Cont'd)

640	1CD3	C90B		CMP	##0B	; CURSOR RT?
650	1CD5	D007		BNE	CKLEFT	
660	1CD7	E009		CPX	##09	; CURSOR=09?
670	1CD9	F0CA		BEQ	CKKEY	
680	1CDB	EB		INX		; X=X+1
690	1CDC	10C7		BPL	CKKEY	
700	1CDE	C90A	CKLEFT	CMP	##0A	; KEY=CURSOR LEFT?
710	1CE0	D006		BNE	CURS	
720	1CE2	CA		DEX		
730	1CE3	10C0		BPL	CKKEY	
740	1CE5	EB		INX		
750	1CE6	10BD		BPL	CKKEY	
760	1CE8	E002	CURS	CPX	##02	; CURSOR L.T. 2?
770	1CEA	B045		BCS	JPNUM	; G.T. 2
780	1CEC	E000		CPX	##00	; CURSOR = 0?
790	1CEE	D02E		BNE	ONE	; =1? BRANCH
800	1CF0	0A		ASL	A	; =0
810	1CF1	0A		ASL	A	; MOVE NUM TO MSNIBBLE
820	1CF2	0A		ASL	A	
830	1CF3	0A		ASL	A	
840	1CF4	4B		PHA		
850	1CF5	A90F		LDA	##0F	; MASK HI NIBBLE
860	1CF7	25AA		AND	##AA	
870	1CF9	85AA		STA	##AA	
880	1CFB	6B		PLA		
890	1CFC	05AA		ORA	##AA	; COMBINE FOR NEW #
900	1CFE	C933	RESUME	CMP	##33	; NUM G.T. 33?
910	1D00	B027		BCS	EROR	
920	1D02	B6C2		STX	##C2	
930	1D04	AA		TAX		
940	1D05	202419		JSR	BCDBIN	
950	1D08	BA		TXA		
960	1D09	0A	MX5	ASL	A	; MULT BY 5
970	1D0A	0A		ASL	A	
980	1D0B	B5C3		STA	##C3	
990	1D0D	BA		TXA		
1000	1D0E	1B		CLC		
1010	1D0F	65C3		ADC	##C3	
1020	1D11	B5EB		STA	##EB	; STORE NEW PT.
1030	1D13	A6C2		LDX	##C2	
1040	1D15	EB	DISA5	INX		; INC CURSOR
1050	1D16	20AF1B		JSR	MOVEL	
1060	1D19	A4EB		LDY	##EB	; RESTORE DAC TABLE PT.
1070	1D1B	4C9A1C		JMP	MLTAB	
1080	1D1E	B5C3	ONE	STA	##C3	; CURSOR = 1, SAVE NUM
1090	1D20	A9F0		LDA	##F0	
1100	1D22	25AA		AND	##AA	
1110	1D24	05C3		ORA	##C3	; COMBINE FOR NEW #
1120	1D26	4CFE1C		JMP	RESUME	
1130	1D29	B5A5	EROR	STA	##A5	; DAC # G.T. 32
1140	1D2B	A9A5		LDA	##A5	
1150	1D2D	B5EB		STA	##EB	
1160	1D2F	30E4		BMI	DISA5	
1170	1D31	20C91E	JPNUM	JSR	NUM	; CURSOR G.T. 2
1180	1D34	4C9A1C		JMP	MLTAB	

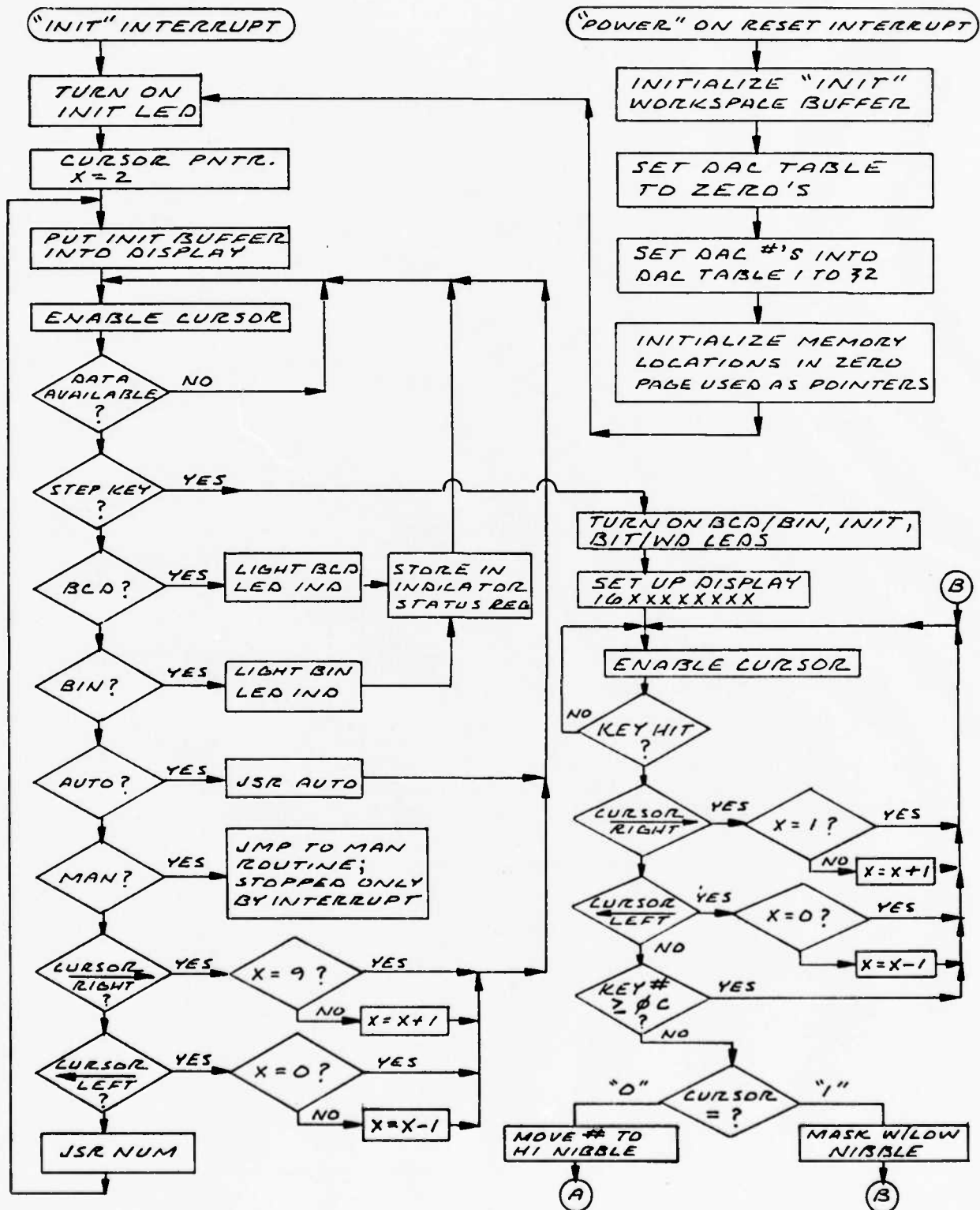


Figure 11A. POWER Flowchart

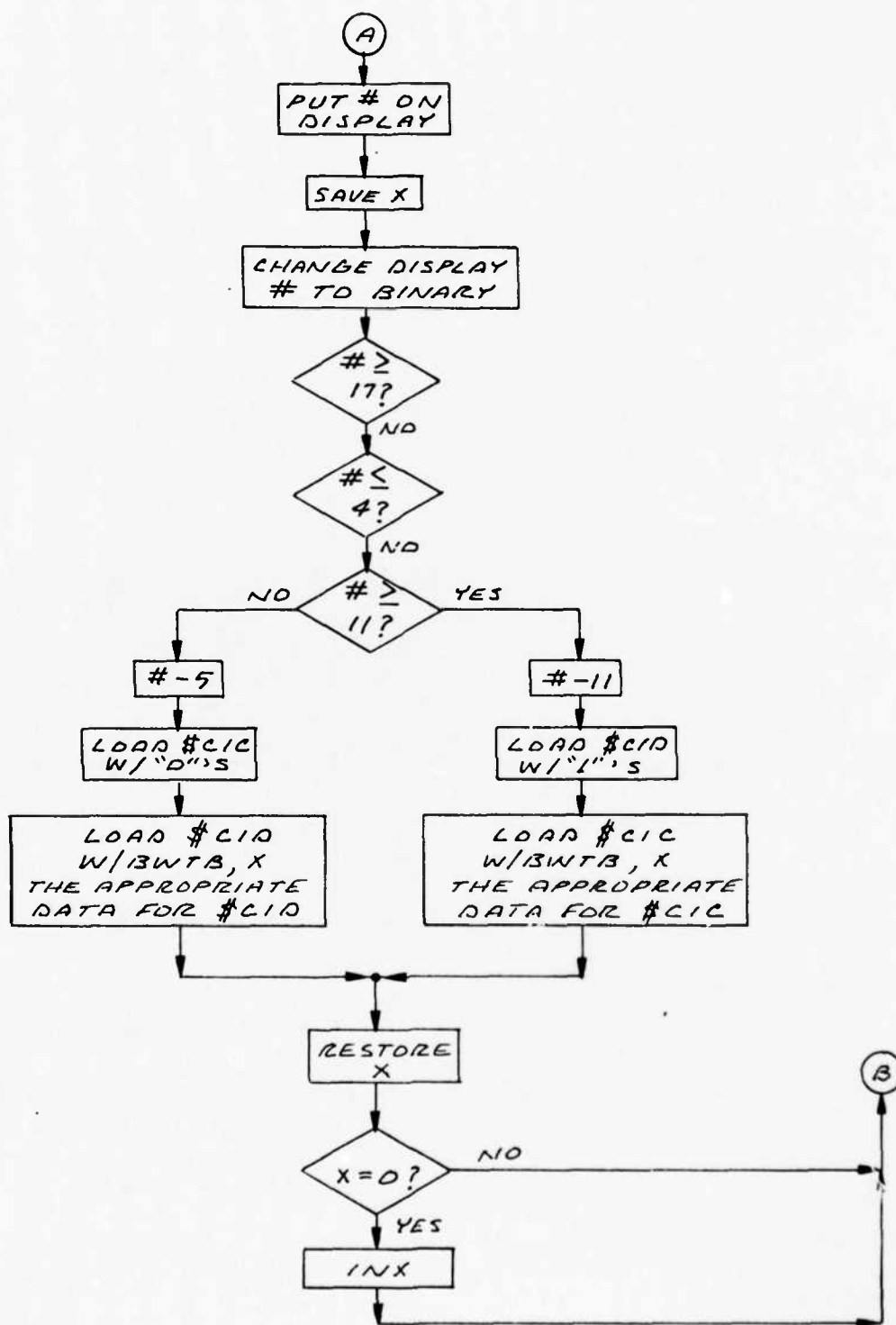


Figure 11B. POWER Flowchart (Cont'd)

Software for POWER Routine (Start)

```

10 0000      ;POWER.....POWER ON RESET ROUTINE
20 0000      ;
30 0000      ;
40 0000      ; 11/6/80
50 0000      AUTO  = $1F00
60 0000      NUM   = $1EC9
70 0000      MOVLIN= $1EF3
80 0000      BCDBIN= $1924
90 0000      MAN    = $1F08
100 0000     KEYBD  = $1FA9
110 0000     SET    = $1C80
120 0000     WDACUP= $1A00
130 0000     DISPX  = $1EBC
140 0000     BWTB   = $1EC0
150 1D40     * = $1D40
160 1D40 A2FF POWER LDX #$FF      ; INIT THE STACK
170 1D42 9A     TXS
180 1D43 58     CLI
190 1D44 DB     CLD
200 1D45
210 1D45 A2FF     LDX #$FF      ; INIT Z-PAGE
220 1D47 A900     LDA #$00      ; TO ALL 00....
230 1D49 85CD     STA $CD      ; INIT STATUS IND BUFFER
240 1D4B 9500     LOOPI STA $00,X
250 1D4D CA      DEX
260 1D4E D0FB     BNE LOOPI
270 1D50 AA      TAX           ; INITIALIZE DAC TABLE
280 1D51 9500     LOOPD STA $00,X ; STORE THE DEC NUM
290 1D53 AB      TAY
300 1D54 BA      TXA
310 1D55 18      CLC
320 1D56 6905     ADC #$05      ; INC THE X POINTER
330 1D58 AA      TAX
340 1D59 98      TYA
350 1D5A FB      SED
360 1D5B 6901     ADC #01      ; INC THE DAC POINTER
370 1D5D DB      CLD
380 1D5E C933     CMP #$33
390 1D60 D0EF     BNE LOOPD
400 1D62 A901     LDA #$01      ; INITIALIZE THE INIT BUFFER
410 1D64 85E0     STA $E0
420 1D66 A916     LDA #$16      ; INITIALIZE B/W BUFFER
430 1D68 85BD     STA $BD
440 1D6A A9FF     LDA #$FF      ; INITIALIZE DATA LEDS
450 1D6C 8D1C0C   STA $C1C
460 1D6F 8D1D0C   STA $C1D
470 1D72
480 1D72          ;
490 1D72          ; INIT ROUTINE
500 1D72 A2FF     INIT LDX #$FF  ; REINIT STACK
510 1D74 9A      TXS
520 1D75 DB      CLD
530 1D76 58      CLI
540 1D77 A908     LDA #08      ; TURN ON INIT LED
550 1D79 05CD     ORA $CD
560 1D7B 8D170C   STA $0C17
570 1D7E A202     LDX #$02      ; SET THE DEC PT POINTER
580 1D80 A0E0     DISP LDY #$E0 ; MOVE THE INIT BUFF TO DISPLAY
590 1D82 20F31E   JSR MOVLIN
600 1D85 BD000C   DECPT LDA $C00,X ; TURN ON DEC PT
610 1D88 AD1010   LDA $1010     ; CHECK FOR BITS/WORD KEY
620 1D8B 0A      ASL A
630 1D8C 0A      ASL A
640 1D8D 0A      ASL A
650 1D8E 900C     BCC BITSWD

```

Software for POWER Routine (Cont'd)

660	1D90	9A	TXS	; SAVE DEC PT
670	1D91	20A91F	JBR KEYBD	; LOOK FOR A KEY
680	1D94	BA	TSX	; GET THE DEC PT
690	1D95	C980	CMP #80	; CK FOR NO KEY
700	1D97	F0EC	BEQ DECPT	
710	1D99	4C461E	JMP BCDKEY	
720	1D9C	A988	LDA #88	; TURN ON B/W, INIT LEDS
730	1D9E	05CD	ORA #CD	
740	1DA0	8D170C	STA #0C17	
750	1DA3	A203	LDX #3	; SET X = 3
760	1DA5	BD8C1E	LDA DISPX,X	; LOAD DISPLAY WITH # AND FF'S
770	1DAB	E8	INX	
780	1DA9	9D100C	STA #0C10,X	
790	1DAC	CA	DEX	
800	1DAD	CA	DEX	
810	1DAE	10F5	BPL BW	
820	1DB0	A5BD	LDA #BD	; SAVE # IN BW BUFFER
830	1DB2	8D100C	STA #0C10	; PUT # ON DISPLAY
840	1DB5	A200	LDX #00	; SET CURSOR POINTER = 0
850	1DB7	8C000C	LDY #C00,X	; ENABLE CURSOR
860	1DBA	9A	TXS	
870	1DBB	20A91F	JSR KEYBD	; CHECK IF KEY HIT
880	1DBE	BA	TSX	
890	1DBF	C980	CMP #80	
900	1DC1	F0F4	BEQ DCWDS	; IF = 0, NO KEY HIT
910	1DC3	C90B	CMP #0B	; CURSOR RIGHT?
920	1DC5	D007	BNE CHEKL	; IF NOT = 0, THEN CHECK LEFT
930	1DC7	E001	CPX #01	; CURSOR ALREADY AT POSITION 1?
940	1DC9	F0EC	BEQ DCWDS	; IF AT 1, DO NOT INC
950	1DCB	E8	INX	; IF AT 0, INC
960	1DCC	10E9	BPL DCWDS	; BRANCH TO CHECK IF KEY HIT
970	1DCE	C90A	CMP #0A	; CURSOR LEFT?
980	1DD0	D007	BNE CHECKC	; IF NOT = 0, THEN CHECK IF KEY >= C
990	1DD2	E000	CPX #00	; CURSOR ALREADY AT POSITION 0?
1000	1DD4	F0E1	BEQ DCWDS	; IF AT 0, DO NOT DEC
1010	1DD6	CA	DEX	; IF AT 1, DEC
1020	1DD7	10DE	BPL DCWDS	; BRANCH TO CHECK IF KEY HIT
1030	1DD9	C90C	CMP #0C	; IF KEY >= C
1040	1DDB	B0DA	BCS DCWDS	; BRANCH TO CHECK IF KEY HIT
1050	1DDD	E000	CPX #00	; MUST BE NUMBER THAT WAS HIT
1060	1DDF	D011	BNE ONES	; CHECK CURSOR POSITION
1070	1DE1	0A	ASL A	; CURSOR AT 0, MOVE TO HI NIBBLE
1080	1DE2	0A	ASL A	
1090	1DE3	0A	ASL A	
1100	1DE4	0A	ASL A	
1110	1DE5	48	PHA	; SAVE ACCUM
1120	1DE6	A90F	LDA #0F	; MASK OFF HI BITS
1130	1DE8	25BD	AND #BD	
1140	1DEA	85BC	STA #BC	
1150	1DEC	68	PLA	
1160	1DED	05BC	ORA #BC	; OR NEW HI ENTRY W/ PREVIOUS LOW
1170	1DEF	4CFC1D	JMP PUTON	
1180	1DF2	48	PHA	; CURSOR AT 1, SAVE ACCUM
1190	1DF3	A9F0	LDA #F0	; MASK OFF LOW BITS
1200	1DF5	25BD	AND #BD	
1210	1DF7	85BC	STA #BC	
1220	1DF9	68	PLA	
1230	1DFA	05BC	ORA #BC	; OR NEW LOW ENTRY W/ PREV. HI
1240	1DFC	86BF	STX #BF	
1250	1DFE	85BD	STA #BD	; PUT IN DISPLAY BUFFER
1260	1E00	8D100C	STA #0C10	; PUT ON DISPLAY
1270	1E03	C911	CMP #11	; CHECK IF NUM >= 11
1280	1E05	B01C	BCS ELEVN	
1290	1E07	C905	CMP #5	; CHECK IF # < 5
1300	1E09	1004	BPL INBND1	
1310	1E0B	A200	LDX #00	; IF # < 5, SET = 0

Software for POWER Routine (Cont'd)

1320	1E0D	F006		BEQ	UNDBND	
1330	1E0F	F8	INBND1	SED		
1340	1E10	38		SEC		
1350	1E11	E904		SBC	##04	; SUBTRACT 4
1360	1E13	D8		CLD		
1370	1E14	AA		TAX		; USE THIS NEW NUMBER AS X PNTR.
1380	1E15	A900	UNDBND	LDA	##00	; STORE 0'S IN \$0C1C
1390	1E17	8D1C0C		STA	\$0C1C	
1400	1E1A	BDC01E		LDA	BWTB,X	
1410	1E1D	8D1D0C		STA	\$0C1D	; STORE APPROPRIATE PATTERN TOO \$0C1D
1420	1E20	4C3C1E		JMP	RESRX	
1430	1E23	C917	ELEVN	CMP	##17	; CHECK IF # > 16
1440	1E25	9004		BCC	INBND2	
1450	1E27	A206		LDX	##06	; SET X POINTER = 6
1460	1E29	D006		BNE	OVRBND	
1470	1E2B	38	INBND2	SEC		
1480	1E2C	F8		SED		
1490	1E2D	E910		SBC	##10	; SUBTRACT 10
1500	1E2F	D8		CLD		
1510	1E30	AA		TAX		; USE THIS NEW NUM AS X PNTR.
1520	1E31	A9FF	OVRBND	LDA	##FF	; STORE 1'S IN \$0C1D
1530	1E33	8D1D0C		STA	\$0C1D	
1540	1E36	BDC01E		LDA	BWTB,X	
1550	1E39	8D1C0C		STA	\$0C1C	; STORE APPROPRIATE PATTERN TO \$0C1C
1560	1E3C	A6BF	RESRX	LDX	\$BF	; RESTORE X
1570	1E3E	E000	NOBW	CPX	##00	; CHECK X = 0
1580	1E40	D001		BNE	MID	; X=1,BRANCH TO CHECK IF KEY HIT
1590	1E42	E8		INX		; X = 0, INC
1600	1E43	4CB71D	MID	JMP	DCWDS	; BRANCH TO CHECK IF KEY HIT
1610	1E46	C90E	BCDKEY	CMP	##0E	; BCD KEY?
1620	1E48	D00A		BNE	CKBIN	
1630	1E4A	A909		LDA	##09	; TURN ON BCD LED
1640	1E4C	8D170C		STA	\$C17	
1650	1E4F	85CD		STA	\$CD	; SAVE IN LED REGISTER
1660	1E51	4C851D		JMP	DECPT	
1670	1E54	C90F	CKBIN	CMP	##0F	; BIN KEY?
1680	1E56	D00A		BNE	CKAUTO	
1690	1E58	A90A		LDA	##0A	; TURN ON BIN LED
1700	1E5A	8D170C		STA	\$C17	
1710	1E5D	85CD		STA	\$CD	; SAVE IN LED REG
1720	1E5F	4C851D		JMP	DECPT	
1730	1E62	C90D	CKAUTO	CMP	##0D	; AUTO KEY?
1740	1E64	D006		BNE	CKMAN	
1750	1E66	20001F		JSR	AUTO	
1760	1E69	4C721D		JMP	INIT	
1770	1E6C	C90C	CKMAN	CMP	##0C	; MANUAL KEY?
1780	1E6E	D003		BNE	CKRIT	
1790	1E70	4C081F		JMP	MAN	; GO TO THE MANUAL ROUTINE
1800	1E73	C90B	CKRIT	CMP	##0B	; CURSOR RT KEY?
1810	1E75	D009		BNE	CKLFT	
1820	1E77	E009		CPX	##09	; LAST CURSOR POS?
1830	1E79	EA		NOP		
1840	1E7A	F001		BEQ	JUMPRT	
1850	1E7C	E8		INX		; INC CURSOR POSITION
1860	1E7D	4C851D	JUMPRT	JMP	DECPT	
1870	1E80	C90A	CKLFT	CMP	##0A	; CURSOR LFT KEY?
1880	1E82	D009		BNE	CKNUM	
1890	1E84	E000		CPX	##00	; CURSOR=1ST POS?
1900	1E86	EA		NOP		
1910	1E87	F001		BEQ	JUMPLF	
1920	1E89	CA		DEX		; DEC CURSOR POSITION
1930	1E8A	4C851D	JUMPLF	JMP	DECPT	
1940	1E8D	20C91E	CKNUM	JSR	NUM	; IF NONE OF THE ABOVE
1950	1E90					; THEN IT MUST BE A NUMBER
1960	1E90	4C801D		JMP	DISP	

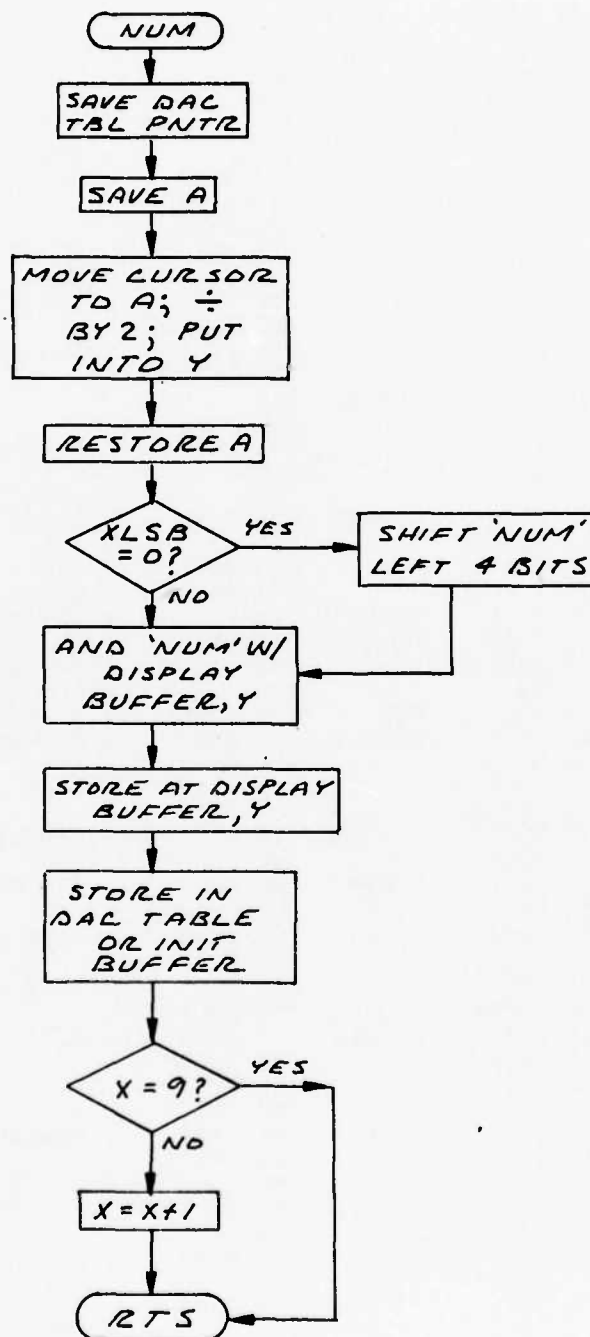


Figure 12. NUM Flowchart

Software for DISPX, NUM, & MOVLIN

```

10 0000      ; DATA TABLES FOR THE INIT-BITS/WORD ROUTINE
20 0000      ; TABLES:  DISPX - CALLED TO BLANK LAST 8
30 0000      ;
40 0000      ;      LEDS OF THE LED DISPLAY
50 0000      ;      BWTB - CALLED TO PLACE APPROPRIATE
60 0000      ;      1'S AND 0'S PATTERNS IN %C1C,%C1D
70 0000      ; 11/6/80
80 0000      ;
90 1EBC      **$1EBC
100 1EBC     ;
110 1EBC FF   DISPX  .BYTE $FF,$FF,$FF,$FF
110 1EBD FF
110 1EBE FF
110 1EBF FF
120 1EC0 00   BWTB  .BYTE $00,$20,$30,$38,$3C,$3E,$3F
120 1EC1 20
120 1EC2 30
120 1EC3 38
120 1EC4 3C
120 1EC5 3E
120 1EC6 3F

10 0000      ; SUBROUTINES:  NUM
20 0000      ;
30 0000      ;
40 0000      ; 11/6/80
50 0000      ;
60 0000      ;
70 0000      ;
80 0000      ;
90 1EC9      **$1EC9
100 1EC9 84C9 NUM    STY %C9      ; SAV THE DAC TAB PT
110 1ECB 48     PHA              ; PUT NUM ON STACK
120 1ECC 8A     TXA              ; PUT CURSOR INTO ACCU
130 1ECD 4A     LSR A            ; DIVIDE BY 2
140 1ECE A8     TAY              ; PUT THIS X/2 INTO Y
150 1ECF 68     PLA              ; GET NUM FROM STACK
160 1ED0 B00C   BCS LSBONE       ; CK CARRY FOR X LND
170 1ED2 0A     ASL A            ; SHIFT NUM INTO MSNIBBLE
180 1ED3 0A     ASL A
190 1ED4 0A     ASL A
200 1ED5 0A     ASL A
210 1ED6 85CA   STA %CA          ; SAVE NUM IN CA
220 1ED8 B1EB   LDA (%EB),Y      ; GET PRESENT DISPLAYED DATA
230 1EDA 290F   AND %0F          ; MASK MSNIBBLE
240 1EDC 1006   BPL STORE
250 1EDE 85CA   LSBONE STA %CA   ; SAVE NUM IN %CA
260 1EE0 B1EB   LDA (%ED),Y      ; GET PRESENT DISPLAYED DATA
270 1EE2 29F0   AND %F0          ; MASK LSNIBBLE
280 1EE4 05CA   STORE  ORA %CA   ; OR OLD DATA WITH NUM
290 1EE6 99AA00 STA %AA,Y        ; STORE IN DISPLAY BUFF
300 1EE9 91EB   STA (%EB),Y      ; STORE IN DAC TABLE
310 1EEB A4C9   LDY %C9          ; RESTORE DAC TAB PT IN Y
320 1EED E009   CPX %09          ; CURSOR=MAX (9) ?
330 1EEF F001   BEQ END
340 1EF1 E8     INX              ; INC CURSOR PT
350 1EF2 60     END    RTS
360 1EF3
370 1EF3
380 1EF3
390 1EF3
400 1EF3
410 1EF3 84ED   MOVLIN STY %EB    ; Y=DAC TAB PT
420 1EF5 A004   LDY %04
430 1EF7 B1ED   MOV    LDA (%EB),Y ; GET DATA
440 1EF9 99100C STA %C10,Y      ; PUT INTO DISPLAY
450 1EFC 88     DEY
460 1EFD 10FB   BPL MOV
470 1EFF 60     RTS

```

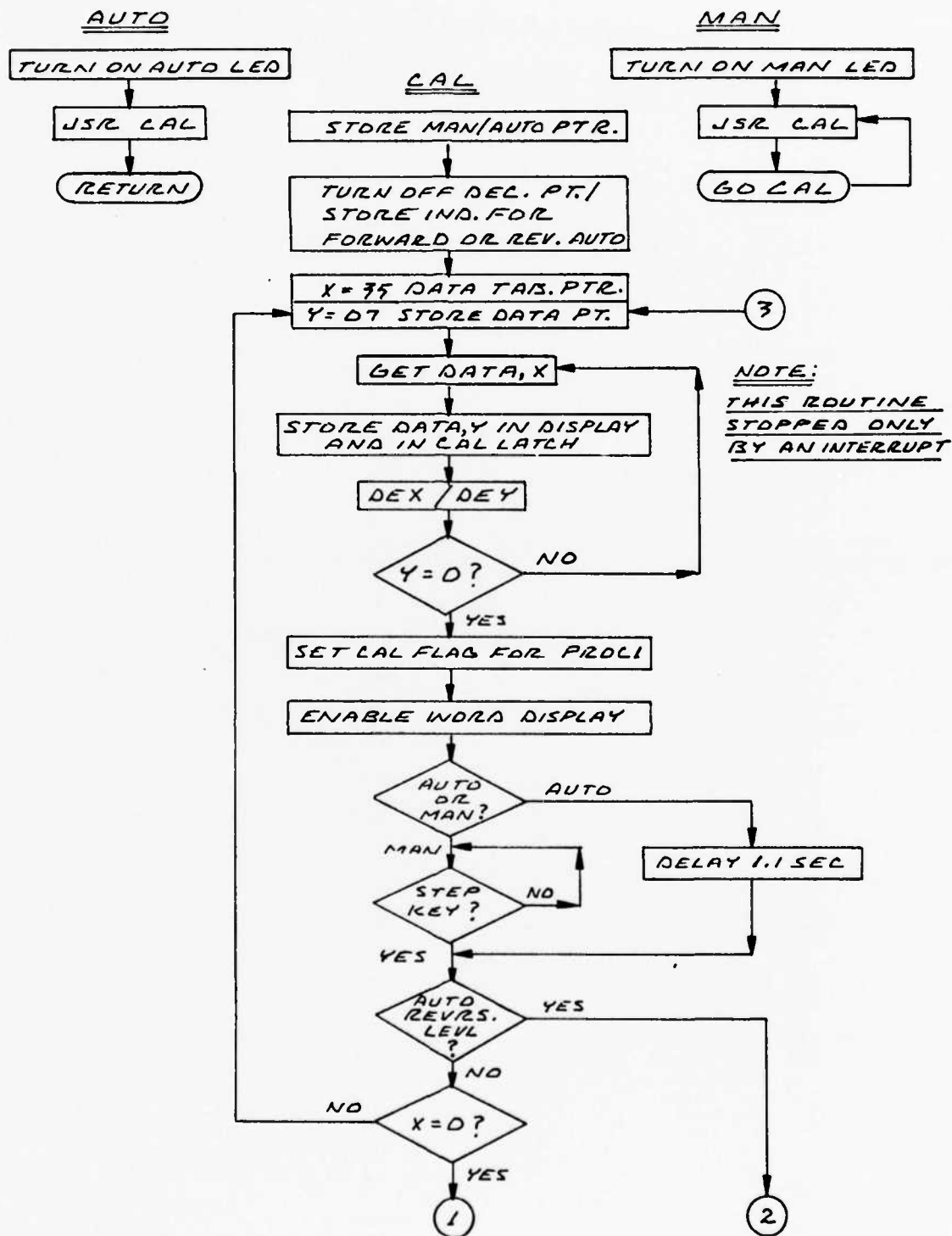
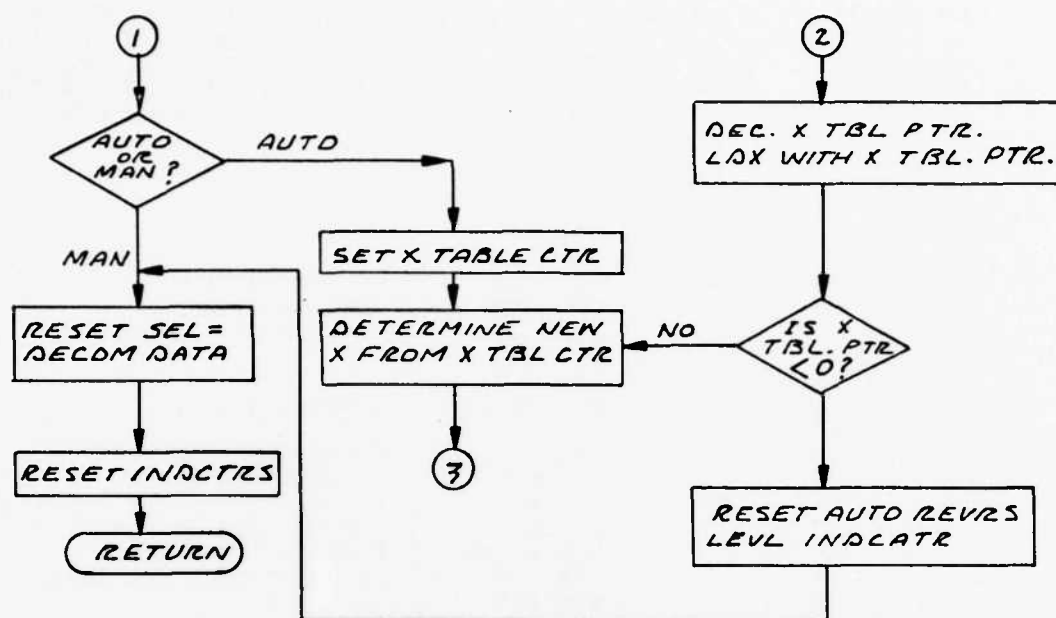


Figure 13A. CALIBRATION Flowchart

Figure 13B. CALIBRATION Flowchart (Cont'd)



AMCAL Subroutine Software

```

10 0000      ; AMCAL
20 0000
30 0000      SUBROUTINES TO CAL THE DACS
40 0000      ; AUTO--AUTOMATIC 5 POINT CAL
50 0000      ; MAN --5 POINT CAL VIA THE STEP KEY AND
60 0000      ; CAN ONLY BE TERMINATED BY THE
70 0000      ; SET OR INIT KEYS....
80 0000      ; 11/6/80
90 0000      ; DATA=$1E95
100 0000     ; JAK=$1EB8
110 1F00     IND = $CD
120 1F00     RUNSTP=$1010
130 1F02 0720 AUTO LDA IND      ; GET IND STATUS
140 1F04 20121F ORA # $20      ; TURN ON AUTO LED
150 1F07 60 JSR CAL          ; RUN THROUGH AUTOMATIC 5 PT CAL
160 1F08 A5CD RTS           ; END
170 1F0A 0910 MAN LDA IND      ; TURN ON MAN LED
180 1F0C 58 ORA # $10
190 1F0D 20121F GOCAL JSR CAL   ; CALIBRATE
200 1F10 10F6 BPL MAN        ; GO BACK AND RECAL
210 1F12 85C9 CAL STA $C9     ; STORE MAN OR AUTO POINTER
220 1F14 8D170C STA $0C17    ; TURN ON MAN OR AUTO LED
230 1F17 8D0A0C STA $0C0A    ; TURN DEC. PT. OFF
240 1F1A A9FF LDA # $FF      ; SET AUTO REV. INDICATOR
250 1F1C 85C7 STA $C7
260 1F1E A900 LDA # $00
270 1F20 8D180C STA $0C18    ; SET SEL=CAL DATA
280 1F23 A222 LDX # $22      ; DATA TABLE POINTER 35DEC
290 1F25 A006 LDY # $06      ; STORE POINTER 7DEC
300 1F27 8D951E LOOPA LDA DATA,X ; MOVE "DATA" TO "STORE"
310 1F2A 99100C STA $0C10,Y
320 1F2D CA DEX
330 1F2E 88 DEY
340 1F2F 10F6 BPL LOOPB
350 1F31 A901 LDA # $01      ; SET CAL FLAG FOR PROC1
360 1F33 8DFF07 STA $7FF
370 1F36 A5C9 LDA $C9        ; GET AUTO-MAN POINTER
380 1F38 2920 AND # $20      ; CK FOR AUTO MODE
390 1F3A D028 BNE AUTDEL     ; GOTO AUTO DELAY
400 1F3C A960 LDA # $60      ; DEBOUNCE APPX. .3 SEC
410 1F3E 206B1F JSR DEL1
420 1F41 AD1010 CKSTEP LDA RUNSTP ; LOOK FOR STEP KEY
430 1F44 0A ASL A
440 1F45 0A ASL A
450 1F46 0A ASL A
460 1F47 B0FB BCS CKSTEP
470 1F49 A5C7 END LDA $C7     ; CHECK FOR AUTO REV LEVL
480 1F4B 1048 BPL REV1
490 1F4D E0FF CPX # $FF      ; END OF DATA?
500 1F4F D0D4 BNE LOOPA
510 1F51 A5C9 LDA $C9        ; IF AUTO, REVERSE CAL LEVELS
520 1F53 2920 AND # $20
530 1F55 D02E BNE REVR5
540 1F57 A910 REST LDA # $10
550 1F59 EA NOP
560 1F5A EA NOP
570 1F5B 8D180C STA $C18
580 1F5E A5CD LDA IND
590 1F60 8D170C STA $0C17
600 1F63 60 RTS
610 1F64 20691F AUTDEL JSR DELAY ; DELAY FOR AUTO INC
620 1F67 F0E0 BEQ END
630 1F69 A9FF DELAY LDA # $FF ; SUBROUTINE TO DELAY 1.1 SEC
640 1F6B 85CA DEL1 STA $CA
650 1F6D 207B1F LOOPC JSR DEL
660 1F70 207B1F JSR DEL

```

AMCAL Software (Cont'd)

```

670 1F73 C6CA          DEC $CA
680 1F75 D0F6          BNE LOOPL
690 1F77 60            RTS
700 1F78 A9FF          DEL      LDA #$FF
710 1F7A B5C8          STA $C8
720 1F7C C6C8          LOOP1    DEC $C8
730 1F7E E6C8          INC $C8
740 1F80 C6C8          DEC $C8
750 1F82 D0F8          BNE LOOP1
760 1F84 60            RTS
770 1F85 A203          REVRS    LDX #$03      ; SET COUNTER FOR X TABLE
780 1F87 B6C7          STX $C7
790 1F89 BD8B1E        CONT     LDA JAK,X      ; DETERMINE NEW X
800 1F8C AA            TAX          ; NEW DATA TABLE POINTER
810 1F8D 1096          BPL LODPA    ; BRANCH TO POINTER STORE
820 1F8F A9FF          REND      LDA #$FF      ; RESET REVERSE LEVELS IND.
830 1F91 B5C7          STA $C7
840 1F93 30C2          BMI REST
850 1F95 C6C7          REV1      DEC $C7
860 1F97 A5C7          LDA $C7      ; CHECK NEW X TABLE POINTER
870 1F99 30F4          BMI REND    ; IF X < 0, JMP TO REND
880 1F9B AA            TAX          ; IF NOT CONTINUE
890 1F9C 10EB          BPL CONT
900 1F9E              ;
910 1E95              ;=$1E95
920 1E95              ;
930 1E95 10          DATA      .BYTE $10,$0F,$FF,$FF,$FF,$FF,$FF
930 1E96 0F
930 1E97 FF
930 1E98 FF
930 1E99 FF
930 1E9A FF
930 1E9B FF
940 1E9C 07          .BYTE $07,$5F,$FF,$FF,$FF,$C0,$00
940 1E9D 5F
940 1E9E FF
940 1E9F FF
940 1EA0 FF
940 1EA1 C0
940 1EA2 00
950 1EA3 05          .BYTE $05,$0F,$FF,$FF,$FF,$80,$00
950 1EA4 0F
950 1EA5 FF
950 1EA6 FF
950 1EA7 FF
950 1EA8 B0
950 1EA9 00
960 1EAA 02          .BYTE $02,$5F,$FF,$FF,$FF,$40,$00
960 1EAB 5F
960 1EAC FF
960 1EAD FF
960 1EAE FF
960 1EAF 40
960 1EB0 00
970 1EB1 00          .BYTE $00,$0F,$FF,$FF,$FF,$00,$00
970 1EB2 0F
970 1EB3 FF
970 1EB4 FF
970 1EB5 FF
970 1EB6 00
970 1EB7 00
980 1EB8 22          JAK        .BYTE $22,$1B,$14,$0D
980 1EB9 1B
980 1EBA 14
980 1EBB 0D

```

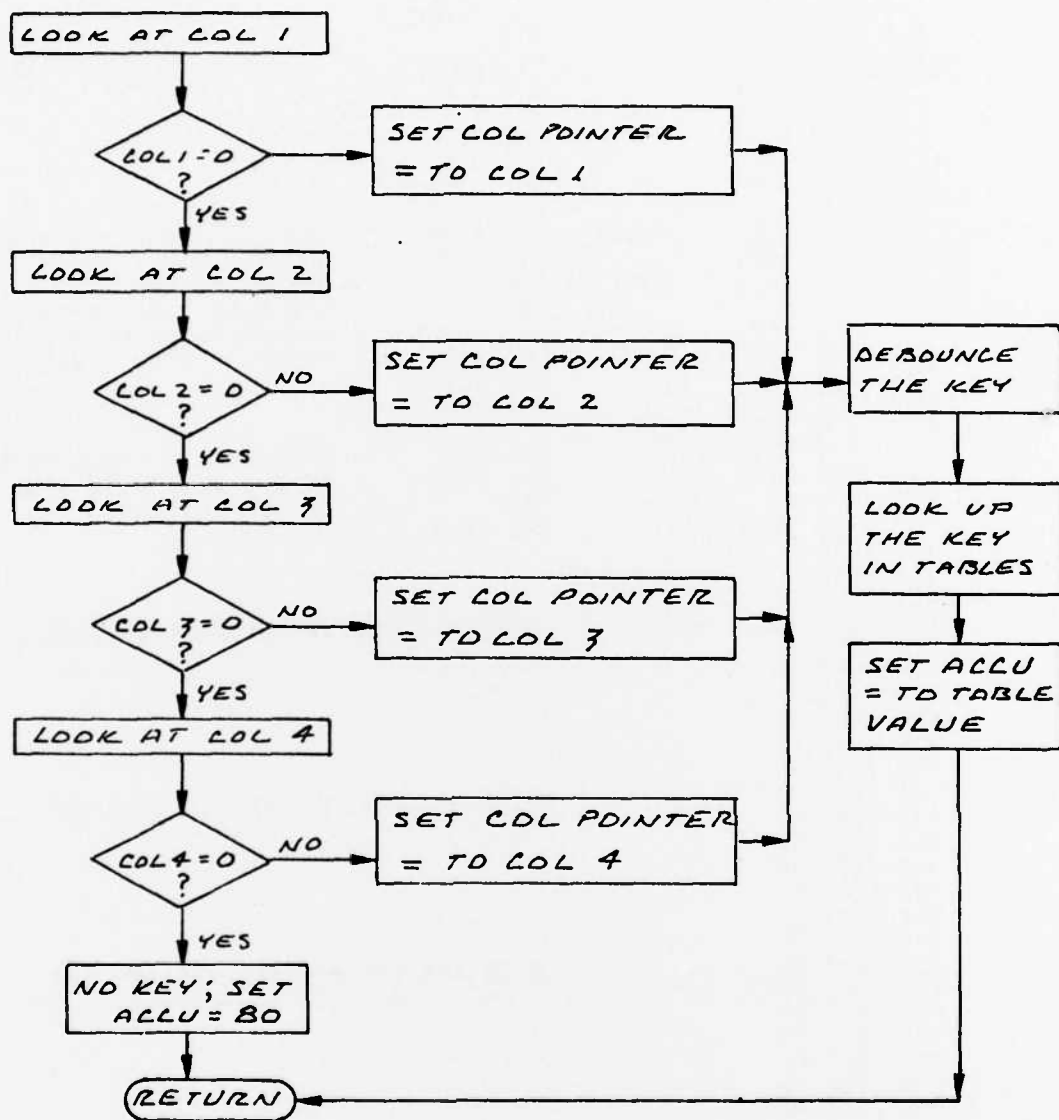


Figure 14. KEYBD Flowchart

Software for KEYBD Subroutine

```

10 0000      : KEYBD
20 0000      : 2/22/83
30 0000      : SUBROUTINE TO SCAN KEYBOARD BY COLUMNS(AB0-AB3)
40 0000      : COL1 =AB0      ROW1 =DB0
50 0000      : COL2 =AB1      ROW2 =DB1
60 0000      : COL3 =AB2      ROW3 =DB2
70 0000      : COL4 =AB3      ROW4 =DB3
80 0000      : DB4-DB7 ARE HELD TO GND.
90 0000      : VALUES RETURNED:
100 0000     : ACCU=KEY VALUE IN HEX
110 0000     : IF NO KEY HIT THEN ACCU=80
120 0000     : X AND Y ARE USED.....
130 0000
140 0000     DEL1  =#1F6B
150 0000     COL1  =#1001
160 0000     COL2  =#1002
170 0000     COL3  =#1004
180 0000     COL4  =#1008
190 0000     *=#1FA9
200 0000
210 1FA9 AD0110 KEYBD LDA COL1      ; CK COL1 FOR KEY DEPRESSED
220 1FAC F004      BEQ NEXT
230 1FAE A000      LDY #000
240 1FB0 101E      BPL WHICH
250 1FB2 AD0210 NEXT LDA COL2      ; CK COL2 KEYS
260 1FB5 F004      BEQ NEXT1
270 1FB7 A004      LDY #004
280 1FB9 D015      BNE WHICH
290 1FBB AD0410 NEXT1 LDA COL3     ; CK COL3 KEYS
300 1FBE F004      BEQ NEXT2
310 1FC0 A008      LDY #008
320 1FC2 D00C      BNE WHICH
330 1FC4 AD0810 NEXT2 LDA COL4     ; CK COL4 KEYS
340 1FC7 F004      BEQ NOKEY
350 1FC9 A00C      LDY #00C
360 1FCB D003      BNE WHICH
370 1FCD A980 NOKEY LDA #80      ; NO KEY...RETURN ACCU=80
380 1FCF 60      RTS
390 1FD0 D8      CLD
400 1FD1 18      CLC
410 1FD2 AA      TAX
420 1FD3 A950      LDA #50      ; KEY DEBOUNCE .3 SEC
430 1FD5 206B1F JSR DEL1
440 1FD8 98      TYA
450 1FD9 7DE11F ADC KTAB,X
460 1FDC AA      TAX
470 1FDD BDEA1F LDA KEYTAB,X
480 1FE0 60      RTS
490 1FE1 00      KTAB .BYTE $00,$00,$01,$00,$02,$00,$00,$00,$03
490 1FE2 00
490 1FE3 01
490 1FE4 00
490 1FE5 02
490 1FE6 00
490 1FE7 00
490 1FE8 00
490 1FE9 03
500 1FEA 0F      KEYTAB .BYTE $0F,$0E,$0D,$0C,$0B,$0A,$09,$08,$07
500 1FEB 0E
500 1FEC 0D
500 1FED 0C
500 1FEE 0B
500 1FEF 0A
500 1FF0 09
500 1FF1 08
500 1FF2 07
510 1FF3 06      .BYTE $06,$05,$04,$03,$02,$01,$00
510 1FF4 05
510 1FF5 04
510 1FF6 03
510 1FF7 02
510 1FF8 01
510 1FF9 00
520 1FFA 80      INTVEC .BYTE $80,$1C,$40,$1D,$72,$1D
520 1FFB 1C
520 1FFC 40
520 1FFD 1D
520 1FFE 72
520 1FFF 1D

```

WDACUP Software (Part 1)

A

```

10 0000 ; WDACUP
20 0000 ; 2/22/83
30 0000 ; ROUTINE TO WRITE PROGRAMS FOR C.RAM AND UPDATE
40 0000 ; BASED ON DAC TABLE ENTERIES.
50 0000 ;
60 0000 ; WDACA -PART A
70 0000 ; WDACB -PART B
80 0000 ;
90 0000 ; POINTERS ;
100 0000 DTABHI=$30
110 0000 BRDTAB=$20
120 0000 HGET =$18
130 0000 LGET =$00
140 0000 HSTORE=$04
150 0000 LSTORE=$00
160 0000 HSTRUP=$02
170 0000 LSTRUP=$00
180 0000 STACK =$2FC2
190 0000 STACKA=$2FC0
200 0000 WISTK =$2FFE
210 0000 ZPADDR=$2FFF
220 0000 IDCRA =$0402
230 0000 IDCRA =$0400
240 0000 MXFRC =$0409
250 0000 FRZ =$2000
260 0000 ;
270 0000 ; SUBROUTINES ;
280 0000 MOVBLK=$1961
290 0000 MBLKUP=$189B
300 0000 MOVEL =$18AF
310 0000 MROUT1=$18BA
320 0000 MROUT2=$18D1
330 0000 RESPT =$1903
340 0000 DACNUM=$1914
350 0000 BCDBIN=$1924
360 0000 DISTAB=$1944
370 0000 MROUT4=$1975
380 0000 TABLE=$1C61
390 0000 CFLAG2=$1B90
400 0000 CKWDCT=$1BF6
410 1A00 *=$1A00
420 1A00
430 1A00 ; WDACUP.....SUB TO WRITE THE
440 1A00 ; C.RAM DAC ENABLE ROUTINE AND
450 1A00 ; THE UPDATE ROUTINE...
460 1A00 A5EB WDACUP LDA $EB
470 1A02 B5ED STA $ED
480 1A04 A900 LDA #0
490 1A06 B5CB STA $CB
500 1A08 B5D2 STA $D2
510 1A0A B5D0 STA $D0
520 1A0C B5D8 STA $D8
530 1A0E B5D9 STA $D9
540 1A10 B5DB STA $DB
550 1A12 B5DD STA $DD
560 1A14 B5E5 STA $E5
570 1A16 B5E7 STA $E7
580 1A18 B5E9 STA $E9
590 1A1A B5EB STA $EB
600 1A1C B5EC STA $EC
610 1A1E A903 LDA #3
620 1A20 B5C6 STA $C6
630 1A22 A901 LDA #1
; DAC TAB PT
; INIT MROUT4 DAC STORE PT

```

WDACA Software (Cont'd)

640	1A24	B3CF		STA \$CF	INIT WDCNT
650	1A26	A930		LDA #DTABHI	INIT DIST TAB PT
660	1A28	B5D1		STA \$D1	
670	1A2A	A210		LDX #\$10	INITIAL 16 PAGES: ALL
680	1A2C	A980		LDA #\$80	LOCATIONS=80
690	1A2E	A000	LOOPB	LDY #\$00	
700	1A30	C6D1		DEC \$D1	
710	1A32	91D0	LOOPA	STA (\$D0),Y	
720	1A34	8B		DEY	
730	1A35	D0FB		BNE LOOPA	
740	1A37	CA		DEX	
750	1A38	D0F4		BNE LOOPB	
760	1A3A	A920		LDA #BRDTAB	SET HI ADDR OF BR DIST TABLE
770	1A3C	B5D7		STA \$D7	
780	1A3E				
790	1A3E	A23F		LDX #\$3F	INIT WI STACK=FF
800	1A40	A9FF		LDA #\$FF	
810	1A42	8DFE07		STA \$07FE	(INIT ID FOR PROC2)
820	1A45	9DC02F	ISTACK	STA STACKA,X	
830	1A48	CA		DEX	
840	1A49	10FA		BPL ISTACK	
850	1A4B				INITIALIZE "GET","STORE","START" ADDRS
860	1A4B	A918		LDA #HGET	
870	1A4D	B5E8		STA \$EB	
880	1A4F	A904		LDA #HSTORE	
890	1A51	B5EA		STA \$EA	
900	1A53	B5E6		STA \$E6	
910	1A55	A902		LDA #HSTRUP	
920	1A57	B5DA		STA \$DA	
930	1A59	B5DC		STA \$DC	
940	1A5B				MOVE ID ROUTINE TO C.RAM
950	1A5B	A000		LDY #\$00	SET "GET" LOW
960	1A5D	A21C		LDX #\$1C	# OF BYTES TO MOVE
970	1A5F	206119		JSR MOVBLK	
980	1A62				MOVE UDATID TO UPDATE ROUTINE
990	1A62	A256		LDX #\$56	SET "GET" LOW
1000	1A64	A213		LDX #\$13	# OF BYTES TO MOVE
1010	1A66	209B18		JSR MBLKUP	
1020	1A69	200319		JSR RESPT	REINIT START PTS.
1030	1A6C				INITIALIZE ID ROUTINES
1040	1A6C	A5E0		LDA \$E0	GET THE ID WORD#
1050	1A6E	8D0204		STA IDCRA	
1060	1A71	A6E4		LDX \$E4	GET MAX FRAMES
1070	1A73	A5CD		LDA \$CD	
1080	1A75	4A		LSR A	
1090	1A76	9004		BCC CON	
1100	1A78	A9FB		LDA #\$FB	
1110	1A7A	D005		BNE CON1	
1120	1A7C	202419	CON	JBR BCDBIN	CONVERT TO BINARY
1130	1A7F	A9DB		LDA #\$DB	
1140	1A81	8D0004	CON1	STA IDCRB	
1150	1A84	B6CC		STX \$CC	
1160	1A86				
1170	1A86	A6E2		LDX \$E2	INIT ID SHIFT CNT
1180	1A88	202419		JBR BCDBIN	CONVERT LSB TO HEX
1190	1A8B	CA		DEX	LSB-1=PRESET COUNT
1200	1A8C	8A		TXA	
1210	1A8D	0910		ORA #\$10	"OR" PRESET WITH SEL=1 (DECOM DATA)
1220	1A8F	8D180C		STA \$C18	
1230	1A92	3B		SEC	
1240	1A93	F8		SED	
1250	1A94	A5E2		LDA \$E2	LSB-MSB=# OF BITS TO BLANK
1260	1A96	E5E1		SBC \$E1	
1270	1A98	AA		TAX	
1280	1A99	8D611C		LDA TABLE,X	
1290	1A9C	8D1B0C		STA \$C18	

WDACA Software (Cont'd)

```

1300 1A9F DB          CLD
1310 1AA0 A903        CKDISP LDA #3      ; SET DAC STORE PT=3
1320 1AA2 85C6        STA #C6
1330 1AA4 A000        LDY #0
1340 1AA6 B1ED        LDA ($ED),Y ; GET DAC# IN DISPLAY
1350 1AA8 F002        BEQ BEGIN
1360 1AAA A905        LDA #5      ; SET = DAC#1
1370 1AAC 85EB        BEGIN STA #EB
1380 1AAE            ; MOVE DAC LINE TO CURRENT
1390 1AAE 20AF18      JSR MOVEL ; LINE BUFFER
1400 1AB1 A5CF        GWDCNT LDA #CF    ; GET WDCNT
1410 1AB3 C5AB        CMP #AB      ; =WD#?
1420 1AB5 F017        BEQ CFLAG1
1430 1AB7 CDFE2F      CMP WISTK ; =WI ON THE STACK?
1440 1ABA D003        BNE CKTAB
1450 1ABC 4CF51A      JMP GETWS
1460 1ABF A5EB        CKTAB LDA #EB    ; GET DAC TAB PT
1470 1AC1 C964        CMP #64      ; =MAX DACS?
1480 1AC3 D003        BNE INCTAB
1490 1AC5 4C901B      JMP CFLAG2
1500 1AC8 D8          INCTAB CLD
1510 1AC9 18          CLC
1520 1ACA 6905        ADC #5      ; INC TABLE PT TO NEXT DAC LINE
1530 1ACC D0DE        BNE BEGIN
1540 1ACE A5D8        CFLAG1 LDA #D8  ; CK FLAG FOR WI
1550 1AD0 F003        BEQ CKWI1
1560 1AD2 4CE61A      JMP CKWI
1570 1AD5 A5AD        CKWI1 LDA #AD   ; WI
1580 1AD7 F0E6        BEQ CKTAB
1590 1AD9 C901        CMP #01    ; WI=01?
1600 1ADB F0E2        BEQ CKTAB
1610 1ADD D02E        BNE CFLAG3
1620 1ADF A980        SETFI LDA #80
1630 1AE1 85D8        STA #D8    ; SET FLAG=FI
1640 1AE3 4CA01A      JMP CKDISP
1650 1AE6 A5AD        CKWI LDA #AD   ; WI
1660 1AEB F003        BEQ CKFI
1670 1AEA 4CBF1A      JMP CKTAB
1680 1AED A5AE        CKFI LDA #AE   ; GET FI
1690 1AEF C901        CMP #01
1700 1AF1 D061        BNE CKFL2
1710 1AF3 F018        BEQ CFLAG3
1720 1AF5 ADFF2F      GETWS LDA ZPADDR ; GET Z-PAGE ADDR FROM STACK
1730 1AF8 85EB        STA #EB    ; MOVE THIS DAC LINE TO BUFFER
1740 1AFA 20AF18      JSR MOVEL
1750 1AFD ADFE2F      LDA WISTK   ; GET NEW WD#
1760 1B00 85AB        STA #AB
1770 1B02 A03D        LDY #3D    ; PUSH DATA TO TOP OF STACK
1780 1B04 B9C02F      PUSHUP LDA STACKA,Y
1790 1B07 99C22F      STA STACK,Y
1800 1B0A 88          DEY
1810 1B0B 10F7        BPL PUSHUP
1820 1B0D 24CB        CFLAG3 BIT #CB  ; FL=YES?
1830 1B0F 3003        BMI CFLAG4
1840 1B11 20BA18      JSR MROUT1 ; PUT IN DATA1
1850 1B14 24D8        CFLAG4 BIT #D8  ; CK FLAG=WI
1860 1B16 3039        BMI GOWDCT
1870 1B18 F8          SED
1880 1B19 18          CLC
1890 1B1A A5CF        LDA #CF      ; GET WORD COUNT
1900 1B1C 65AD        ADC #AD      ; WDCNT + WI =NEW WI
1910 1B1E B031        BCS GOWDCT ; IF >=100 THEN DON'T PUT ON STACK
1920 1B20 85C4        STA #C4    ; SAVE NWI
1930 1B22 D8          CLD
1940 1B23 A03E        LDY #3E
1950 1B25 A5C4        SCNSTK LDA #C4 ; GET NWI

```

WDACA Software (Cont'd)

1960	1B27	D9C02F		CMP STACKA,Y ; CMP WITH DATA ON STACK
1970	1B2A	F025		BEQ GOWDCT ; =NWI?
1980	1B2C	9006		BCC PUSH ; STACK>NWI?
1990	1B2E	88		DEY ; LOOK AT NEXT WD IN STACK
2000	1B2F	88		DEY
2010	1B30	10F3		BPL SCNSTK
2020	1B32	301D		BMI GOWDCT
2030	1B34	C8	PUSH	INX
2040	1B35	C8		INX
2050	1B36	84C5		STY \$C5 ; SAVE POINTER
2060	1B38	A000		LDY #\$00
2070	1B3A	B9C22F	DOWN	LDA STACK,Y ; PUSH STACK DOWN
2080	1B3D	99C02F		STA STACKA,Y
2090	1B40	C8		INX
2100	1B41	C4C5		CPY \$C5
2110	1B43	30F5		BMI DOWN ; Y<POINTER?
2120	1B45	88		DEY
2130	1B46	A5EB		LDA \$EB ; GET DAC TAB ADDR
2140	1B48	99C02F		STA STACKA,Y ; PUT ON STACK
2150	1B4B	88		DEY
2160	1B4C	A5C4		LDA \$C4
2170	1B4E	99C02F		STA STACKA,Y ; PUT ON STACK
2180	1B51	4CF61B	GOWDCT	JMP CKWDCT ; GO CK WDCNT
2190	1B54	24CB	CKFL2	BIT \$CB ; FL=YES?
2200	1B56	1005		BPL PUTD2
2210	1B58	207519		JSR MR0UT4 ; PUT IN DATA4
2220	1B5B	1007		BPL DIST
2230	1B5D	20D118	PUTD2	JSR MR0UT2 ; PUT ROUTINES DATA2 & UDAT2
2240	1B60	A980		LDA #\$80
2250	1B62	85CB		STA \$CB ; SET FL=YES
2260	1B64	A6AC	DIST	LDX \$AC ; GET FR#
2270	1B66	86D5		STX \$D5 ; SAVE IN NFI
2280	1B68	204419		JSR DISTAB ; CAL BR DIST ADDR
2290	1B6B	A5AE		LDA \$AE ; GET FI

WDACB Software (WDACUP, Part 2)

A

```

10 0000      : WDACUP      .....CONTINUATION
20 0000      : 2/22/83
30 0000      : ROUTINE TO WRITE PROGRAMS FOR C.RAM AND UPDATE
40 0000      : BASED ON DAC TABLE ENTERIES.
50 0000      :
60 0000      : WDACB      -PART B
70 0000      : WDACA      -PART A
80 0000      :
90 0000      : POINTERS :
100 0000     DTABHI=$30
110 0000     BRDTAB=$20
120 0000     HGET  =$18
130 0000     LGET  =$00
140 0000     HSTORE=$04
150 0000     LSTORE=$00
160 0000     HSTRUP=$02
170 0000     LSTRUP=$00
180 0000     STACK =$2FC2
190 0000     STACKA=$2FC0
200 0000     WISTK  =$2FFE
210 0000     ZPADDR=$2FFF
220 0000     IDCRA  =$0402
230 0000     IDCRB  =$0400
240 0000     MXFRC  =$0409
250 0000     FRZ    =$2000
260 0000     :
270 0000     : SUBROUTINES :
280 0000     MOVBLK=$1961
290 0000     MBLKUP=$189B
300 0000     MOVEL  =$18AF
310 0000     MROUT1=$18BA
320 0000     MROUT2=$18D1
330 0000     RESPT  =$1903
340 0000     DACNUM=$1914
350 0000     BCDBIN=$1924
360 0000     DISTAB=$1944
370 0000     MROUT4=$1975
380 0000     TABLE=$1C61
390 0000     CFLAG2=$1B90
400 0000     CKWDCT=$1BF6
410 0000     CKTAB  =$1ABF
420 0000     SETFI  =$1ADF
430 0000     CKDISP=$1AA0
440 0000
450 1B6D     *=$1B6D
460 1B6D
470 1B6D
480 1B6D
490 1B6D
500 1B6D F016
510 1B6F 18
520 1B70 F8
530 1B71 A5D5
540 1B73 65AE
550 1B75 B00E
560 1B77 85D5
570 1B79 D8
580 1B7A AA
590 1B7B C5E4
600 1B7D B006
610 1B7F 204419
620 1B82 4C6F1B
630 1B85 18
640 1B86 D8

      BEQ BRDST5
      CLC
      SED
      LDA $D5
      ADC $AE
      BCS BRDST5
      STA $D5
      CLD
      TAX
      CMP $E4
      BCS BRDST5
      JSR DISTAB
      JMP LOOPFI
      CLC
      CLD

      : WDACUP.....SUB TO WRITE THE
      : C.RAM DAC ENABLE ROUTINE AND
      : THE UPDATE ROUTINE...
      :
      : GO INC BR DIST BY 5
      :
      : SET DEC MODE
      : NFI=NFI+FI
      :
      : IF NFI>=100 THEN BRANCH
      : SAVE IN NFI
      :
      : COMPARE WITH MAX FRAMES
      : PUT IN DIST TAB
      : BRDIST=BRDIST+5

```

WDACB Software (Cont'd)

650	1887	A5DD		LDA \$DD	: GET BR COUNTER
660	1889	6903		ADC #5	
670	188B	85DD		STA \$DD	
680	188D	4CBF1A		JMP CKTAB	
690	1890	24D8	CFLAG2	BIT \$D8	: CK FOR FI
700	1892	3003		BMI CKBRD	
710	1894	4CDF1A		JMP SETFI	
720	1897	A5DD	CKBRD	LDA \$DD	: BRDIST=07
730	1899	F05B		BEQ CKWDCT	
740	189B	D8		CLD	: PUT IN BR TO NEXT ROUTINE DIST
750	189C	38		SEC	
760	189D	A5E9		LDA \$E9	: STORE-START
770	189F	E5E5		SBC \$E5	
780	1BA1	F016		BEQ CLDIST	
790	1BA3	38		SEC	
800	1BA4	E904		SBC #4	
810	1BA6	A8		TAY	
820	1BA7	A903		LDA #3	: INIT DIST
830	1BA9	91E5	STORE1	STA (\$E5),Y	: PUT DIST INTO ROUTINE
840	1BAB	18		CLC	
850	1BAC	6905		ADC #5	
860	1BAE	AA		TAX	
870	1BAF	98		TYA	
880	1BB0	38		SEC	
890	1BB1	E905		SBC #5	: DEC Y PT
900	1BB3	3004		BMI CLDIST	
910	1BB5	A8		TAY	
920	1BB6	8A		TXA	
930	1BB7	10F0		BPL STORE1	
940	1BB9	A264	CLDIST	LDX #\$64	: SET REST OF COL=NEXT BR DIST
950	1BBB	A000		LDY #\$00	
960	1BBD	A920		LDA #BRDTAB	: INIT HI ADDR
970	1BBF	85D7		STA \$D7	
980	1BC1	A5D2		LDA \$D2	: SET LOW ADDR = COLCNT
990	1BC3	85D6		STA \$D6	
1000	1BC5	CA	LOOPD	DEX	: CK FOR END OF COLUMN
1010	1BC6	D002		BNE TEST	
1020	1BC8	F016		BEQ PUTBR	
1030	1BCA	B1D6	TEST	LDA (\$D6),Y	: GET DATA IN DIST TAB
1040	1BCC	1004		BPL NXTCOL	: IF=80 THEN GOTO NEXT COL
1050	1BCE	A5DD		LDA \$DD	: GET NEXT BR DIST
1060	1BD0	91D6		STA (\$D6),Y	: PUT DIST INTO TABLE
1070	1BD2	D8	NXTCOL	CLD	
1080	1BD3	18		CLC	
1090	1BD4	A920		LDA #\$20	: ADD 32 TO LOW ADDR TO
1100	1BD6	65D6		ADC \$D6	: OBTAIN NEXT ADDR IN COL
1110	1BD8	85D6		STA \$D6	
1120	1BDA	90E9		BCC LOOPD	
1130	1BDC	E6D7		INC \$D7	: INC IF PAGE IS CROSSED
1140	1BDE	10E5		BPL LOOPD	
1150	1BE0	A000	PUTBR	LDY #0	: PUT IN NOP8
1160	1BE2	A9EA		LDA #0EA	: TO EVEN UP BR DIST IN C.RAM
1170	1BE4	18		CLC	
1180	1BE5	91E9		STA (\$E9),Y	
1190	1BE7	E6E9		INC \$E9	
1200	1BE9	91E9		STA (\$E9),Y	
1210	1BEB	E6E9		INC \$E9	
1220	1BED	9002		BCC RESET	
1230	1BEF	E6EA		INC \$EA	: INC FOR PAGE CROSSING
1240	1BF1	200319	RESET	JSR RESPT	
1250	1BF4	E6D2		INC \$D2	: INC COL CNT
1260	1BF6	E6CF	CKWDCT	INC \$CF	: INC WORD CNT
1270	1BF8	A5CF		LDA \$CF	: DOES WDCNT=MAX WDS?
1280	1BFA	C5E3		CMP \$E3	
1290	1BFC	F00B		BEQ MROUTE	
1300	1BFE	A900		LDA #\$00	

WDACB Software (Cont'd)

```

1310 1C00 85D8          STA $D8      ; SET FLAG=WI
1320 1C02 85CB          STA $CB      ; SET FL=NO
1330 1C04 85DD          STA $DD      ; SET BR DIST=0
1340 1C06 4CA01A        JMP CKDISP
1350 1C09 A20D          MROUTE LDX $0D      ; PUT IN DATEND FOR MAIN
1360 1C0B A044          LDY $44      ; ENABLE ROUTINE
1370 1C0D 206119        JSR MOVBLK
1380 1C10 A20C          LDX $0C      ; PUT IN UDATID
1390 1C12 A07A          LDY $7A
1400 1C14 209B18        JSR MBLKUP
1410 1C17 D8           CLD
1420 1C18 38           SEC
1430 1C19 A920          LDA $20      ; CK TO SEE HOW MANY COL WERE USED
1440 1C1B E5D2          SBC $D2      ; 32-COLCNT=COLUMNS USED
1450 1C1D A003          LDY $03
1460 1C1F 91DB          STA ($D8),Y ; PUT INTO UDATED
1470 1C21
1480 1C21
1490 1C21
1500 1C21
1510 1C21 A200          LDX $00      ; RESET COLCNT=0
1520 1C23 86D2          STX $D2
1530 1C25 A6E4          LDX $E4      ; GET MAX FRAMES
1540 1C27 204419        JSR DISTAB ; CAL ADDR TO STORE
1550 1C2A A01F          LDY $1F
1560 1C2C B90020        LOOPF  LDA FRZ,Y ; GET FR#0 DATA
1570 1C2F 91D0          STA ($D0),Y ; STORE AT MAX FRAMES*32
1580 1C31 88           DEY
1590 1C32 10F8          BPL LOOPF
1600 1C34 A5CD          LDA $CD      ; GET INDICATOR STATUS
1610 1C36 2903          AND $3
1620 1C38 85CD          STA $CD      ; GET BCD OR BIN ONLY
1630 1C3A 8D0A0C        STA $C0A    ; TURN OFF DECIMAL PT
1640 1C3D 0940          ORA $40      ; INIT DATA LED
1650 1C3F 85C0          STA $C0
1660 1C41 A5CD          LDA $CD
1670 1C43 0980          ORA $80      ; INIT WAIT LED
1680 1C45 85C1          STA $C1
1690 1C47 8D170C        STA $C17
1700 1C4A A200          LDX $00
1710 1C4C 86E9          STX $E9      ; INIT "LOW" UPDATE TABPT
1720 1C4E A920          LDA $20
1730 1C50 8DFF07        STA $7FF    ; START PROC1
1740 1C53 D8           CLD
1750 1C54 A5CD          LDA $CD
1760 1C56 2903          AND $03
1770 1C58 85CD          STA $CD
1780 1C5A 8D0A0C        STA $0C0A
1790 1C5D 4C691C        JMP $1C69
1800 1C60 FF           .BYTE $FF
1810 1C61 00          TABLE .BYTE 0,0,0,0,$10,$30,$70,$F0
1810 1C62 00
1810 1C63 00
1810 1C64 00
1810 1C65 10
1810 1C66 30
1810 1C67 70
1810 1C68 F0
1820 1C69 0940          ORA $40
1830 1C6B 8D170C        STA $0C17
1840 1C6E ECFE07        WAIT  CPX $7FE
1850 1C71 D0FB          BNE WAIT
1860 1C73 4C0702        JMP $207

```

Software for SUBRT (WDACUP Subroutines)

A

```

10 0000      ; SUBRT....SUBROUTINES FOR WDACUP
20 0000      ; 7/30/80
30 0000      ;*****
40 0000      ; DATA FOR PROCESSOR 1
50 0000      ;*****
60 1800      *=$1800
70 1800      BRDTAB=$20
80 1800      SUBS  = $189B
90 1800      SUB8A = $183F
100 1800     ;
110 1800     FB      DATAID SED      ; SET DECIMAL OR BIN MODE
120 1801     A201     LDX  #001      ; LOOK FOR WORD #01
130 1803     EC010B   MAJSYN CPX  #001 ; FIND MINOR FRAME ID
140 1806     D0FB     BNE MAJSYN
150 1808     AD000B   LDA  #000      ; GET THE ID
160 180B     F00C     BEQ 8AVE
170 180D     D0F4     BNE MAJSYN ; IF NOT = 00 GO BACK
180 180F     EC010B   MINSYN CPX  #001
190 1812     D0FB     BNE MINSYN ; THIS IS MINOR FRAME SYNC
200 1814     CD000B   CMP  #000      ; IS THE ID = ID COUNTER
210 1817     D0EA     BNE MAJSYN ; IF NOT CORRECT THEN RESTART
220 1819     8DFE07   SAVE  STA  #7FE ; SAVE ID FOR PROC2
230 181C     ;
240 181C     ; DATA TO ENABLE A DAC EVERY MINOR FRAME....
250 181C     ;
260 1825     *=$1825
270 1825     ;
280 1825     A000     DATA1 LDY  #000 ; WORD# (FROM PROC2)
290 1827     CC010B   LOOP1  CPY  #001 ; GET WORD# FROM DECOM
300 182A     D0FB     BNE LOOP1
310 182C     8D000C   STA  #0C00 ; ENABLE A DAC (FROM PROC2)
320 182F     ;
330 182F     ; DATA TO ENABLE A DAC ON PARTICULAR MINOR FRAME...
340 182F     ;
350 182F     A000     DATA2 LDY  #000 ; WORD# (FROM PROC2)
360 1831     CC010B   LOOP2  CPY  #001 ; GET WORD# FROM DECOM
370 1834     D0FB     BNE LOOP2
380 1836     F000     BEQ NEXTD ; BR DIST IS FROM UPDATE ROUTINE
390 1838     8D000C   NEXTD  STA  #0C00 ; ENABLE A DAC (FROM PROC2)
400 183B     ; NEXTD DEPENDS ON UPDATE ROUTINE.. BVC
410 183B     ; MAY BR TO NEXT STATEMENT--DEPENDING
420 183B     ; ON THE UPDATE TABLE....
430 183F     *=$SUBSA
440 183F     ;
450 183F     ; DATA TO ENABLE MULT DACS/WORD
460 183F     ; USED WITH DATA2 ROUTINE...
470 183F     ;
480 183F     F000     DATA4 BEQ NEXTE ; BR DIST (FROM UPDATE)
490 1841     8D000C   NEXTE  STA  #0C00 ; ENABLE THE DAC OR SKIP TO
500 1844     ; TO NEXT INSTRUCTION..DEPENDS ON UPDATE.
510 1844     ;
520 1844     ; DATA USED TO LOOP BACK TO A NEW MINOR FRAME
530 1844     ; TO NEW MAJOR FRAME....
540 1844     ;
550 1844     STARTR=$0403 ; AN ADDRESS IN C.RAM
560 1844     FRCNTR=$040F ; AN ADDRESS IN C.RAM
570 1844     18      DATEND CLC
580 1845     6901     ADC  #001 ; INC THEN FRAME COUNTER
590 1847     C5CC     CMP  #CC ; COMPARE WITH MAX FRAMES
600 1849     D003     BNE MFR ; END OF MAJOR FRAME??
610 184B     4C0304   JMP 8STARTR ; GO GET A NEW MAJOR FRAME
620 184E     4C0F04   MFR    JMP FRCNTR ; GO GET A NEW MINOR FRAME
630 1851     ;

```

Software for UPDATE (WDACUP)

```

640 1851 ;*****
650 1851 ; DATA FOR PROC2..UPDATE
660 1851 ;*****
670 1851 ;
680 1851 ; DATA TO WRITE UPDATE ROUTINE...THE WDACUP ROUTINE
690 1851 ; WILL MOVE THIS DATA TO 200.
700 1851 ;
710 1856 X=$1856
720 1856 ECFE07 BEGIN1 CPX $7FE ; DOES X=ID
730 1859 9006 BCC SETX ; IF ID > X THEN GO SET X
740 185B F0F9 BEQ BEGIN1 ; IF ID=X THEN WAIT
750 185D A020 MJSYNC LDY #$20 ; INIT "LOW" TAB PT
760 185F 84EA STY $EA ; INIT "HI" TAB PT
770 1861 AEFE07 SETX LDX $7FE ; SET X=ID
780 1864 ;
790 1864 ; ROUTINE TO UPDATE PROCESSOR1 BRANCH INST.
800 1864 ;
810 186D X=$186D
820 186D CRADD = $400
830 186D A900 UDAT2 LDA #$00 ; FIND WORD#(PUT IN BY WMENAB)
840 186F CD000B LOOPU2 CMP $800 ; CMP WITH DECOM ADDR LINES
850 1872 B0FB BCS LOOPU2 ; IF WORD#> DECOM ADDR THEN:
860 1874 B1E9 LDA ($E9),Y ; GET BRANCH DIST FROM DISTAB
870 1876 BD0004 STA CRADD ; STORE IN SOME CRAM ADDR
880 1879 CB INY ; INC POINTER FOR TABLE
890 187A ;
900 187A ; ROUTINE TO END THE UPDATE ROUTINE
910 187A ; CHECKS FOR END OF MAJOR FRAME
920 187A ;
930 187A BEGIN = $200
940 187A 98 UPDATED TYA ; GET UPDATE TAB PT
950 187B 18 CLC
960 187C 6900 ADC #$00 ; ADD # OF INC (FROM WDAC)
970 187E AB TAY
980 187F D002 BNE CKMJ
990 1881 E6EA INC $EA ; INC HI TAB PT (IF Y=0)
1000 1883 4C0002 CKMJ JMP BEGIN
1010 1886 ;
1020 1886 ;*****
1030 1886 ; SUBROUTINES FOR WDACUP.....
1040 1886 ;*****
1050 1886 ; 5/23/80
1060 189B X=SUBB
1070 189B ;
1080 189B ; MBLKUP.....SUB TO MOVE A BLK
1090 189B 84E7 MBLKUP STY $E7 ; OF DATA TO UPDATE ROUTINE
1100 189D A000 LDY #$00 ; Y=START ADDR LOW "GET"
1110 189F B1E7 LOOPU LDA ($E7),Y ; "GET"
1120 18A1 91D9 STA ($D9),Y ; "STORE"
1130 18A3 E6E7 INC $E7
1140 18A5 E6D9 INC $D9
1150 18A7 D002 BNE CONT
1160 18A9 E6DA INC $DA
1170 18AB CA CONT DEX
1180 18AC D0F1 BNE LOOPU
1190 18AE 60 RTS
1200 18AF ;
1210 18AF ; MOVEL.....MOVE A LINE TO
1220 18AF A004 MOVEL LDY #4 ; DISPLAY BUFFER.
1230 18B1 B1EB LOOPL LDA ($EB),Y ; GET DATA
1240 18B3 99AA00 STA $AA,Y ; STORE IN BUFFER
1250 18B6 BB DEY
1260 18B7 10FB BPL LOOPL
1270 18B9 60 RTS
1280 18BA ;
1290 18BA ; ROUTINES TO MOVE THE DATA TO C. RAM
1300 18BA ; AND TO THE UPDATE ROUTINE. THESE ROUTINES
1310 18BA ; PUT IN THE DAC#, WD# AND INTERVALS IF NEEDED.
1320 18BA ;
1330 18BA ;

```

UPDATE Software (Cont'd)

1340	18BA				; MROUT1..PUTS IN FI ROUTINE
1350	18BA	A20A	MROUT1	LDX #0A	; BYTES TO MOVE
1360	18BC	A025		LDY #25	; START ADDR LOW
1370	18BE	206119		JSR MOVBLK	
1380	18C1	A5AB		LDA #AB	; WD#
1390	18C3	A001		LDY #01	
1400	18C5	91E5		STA (#E5),Y	
1410	18C7	201419		JSR DACNUM	; GET BCD DAC#
1420	18CA	A008		LDY #08	
1430	18CC	91E5		STA (#E5),Y	; PUT IN DAC #
1440	18CE	4C0319		JMP RESPT	; RESET POINTERS
1450	18D1				
1460	18D1				
1470	18D1	A20C	MROUT2	LDX #0C	; MROUT2..PUTS IN DATA2 & UDAT2
1480	18D3	A02F		LDY #2F	; # BYTES TO MOVE
1490	18D5	206119		JSR MOVBLK	; START LOC
1500	18D8	A20D		LDX #0D	; MOVE DATA2
1510	18DA	A06D		LDY #6D	; # BYTES TO MOVE
1520	18DC	209B18		JSR MBLKUP	; START
1530	18DF	A001		LDY #01	; MOVE UDAT2
1540	18E1	A5AB		LDA #AB	; PUT WD# INTO ROUTINES
1550	18E3	91DB		STA (#DB),Y	; IN UDAT2
1560	18E5	91E5		STA (#E5),Y	; IN DATA2
1570	18E7	201419		JSR DACNUM	
1580	18EA	A00A		LDY #0A	
1590	18EC	91E5		STA (#E5),Y	; PUT DAC# INTO ROUTINE
1600	18EE	18		CLC	; DATA2 INTO UDAT2
1610	18EF	D8		CLD	
1620	18F0	A5E5		LDA #E5	; LOW "START" ADDR
1630	18F2	6908		ADC #8	; ACCU=BR DIST ADDR
1640	18F4	91DB		STA (#DB),Y	; STORE IN UDAT2
1650	18F6	9006		BCC CLEAR	; BEYOND PAGE?
1660	18F8	AA		TAX	; IF CARRY=1 THEN ADD WENT NEXT PAGE
1670	18F9	CA		DEX	; DOES DEX=FF? THEN NBIT=1
1680	18FA	1002		BPL CLEAR	
1690	18FC	E6E6		INC #E6	; POINT TO NEXT PAGE
1700	18FE	C8	CLEAR	INY	; INC "STORE" POINTER
1710	18FF	A5E6		LDA #E6	; GET "HI STORE" LOC
1720	1901	91DB		STA (#DB),Y	; PUT INTO DATA2
1730	1903				
1740	1903	A5E9	RESPT	LDA #E9	; RESET PTS FOR "START" ADDR
1750	1905	85E5		STA #E5	
1760	1907	A5EA		LDA #EA	
1770	1909	85E6		STA #E6	
1780	190B	A5D9		LDA #D9	
1790	190D	85DB		STA #DB	
1800	190F	A5DA		LDA #DA	
1810	1911	85DC		STA #DC	
1820	1913	60		RTS	
1830	1914				
1840	1914				
1850	1914	A6AA	DACNUM	LDX #AA	; CONVERT DAC# TO AN ADDRESS (8 BITS)
1860	1916	202419		JSR BCDBIN	; GET BCD DAC#
1870	1919	CA		DEX	; CONVERT TO HEX
1880	191A	8A		TXA	
1890	191B	A6ED		LDX #ED	; CMP CURRENT LINE IN DISPLAY
1900	191D	E4EB		CPX #EB	; WITH DAC TABLE PT
1910	191F	D002		BNE DACEND	
1920	1921	0940		ORA #40	; SET BIT 6 SO WORD DISPLAY
1930	1923	60	DACEND	RTS	; LEDS WILL BE LATCHED.
1940	1924				
1950	1924				
1960	1924				
1970	1924				
1980	1924				
1990	1924				
2000	1924	A950	ADDN	=#D3	
2010	1926	85D3	BIN	=#D4	
2020	1928	8A	BCDBIN	LDA #50	; # TO BE ADDED TO BIN#
2030	1929	290F		STA ADDN	
				TXA	
				AND #0F	; KEEP LOWER NIBBLE

UPDATE Software (Cont'd)

2040	192B	85D4		STA BIN	;	SAVE BIN#
2050	192D	8A		TXA	;	PUT BCD# IN ACCU
2060	192E	A204		LDX #04		
2070	1930	2A	ROTAT	ROL A	;	CK FOR CARRY
2080	1931	AB		TAY	;	SAVE #
2090	1932	9007		BCC NOADD		
2100	1934	18		CLC		
2110	1935	A5D4		LDA BIN	;	ADD ADDN TO BIN
2120	1937	65D3		ADC ADDN		
2130	1939	85D4		STA BIN		
2140	193B	46D3	NOADD	LSR ADDN	;	READY ADDN FOR NEXT ADD
2150	193D	98		TYA	;	GET SHIFTED BCD#
2160	193E	CA		DEX		
2170	193F	D0EF		BNE ROTAT		
2180	1941	A6D4		LDX BIN	;	RETURN BIN# IN X
2190	1943	60		RTS		
2200	1944					
2210	1944					
2220	1944					DISTAB..SUB TO CALCULATE AN ADDR
2230	1944					IN THE DISTANCE TABLE FOR UPDATE.
2250	1944					X=FRAME #
2260	1944	202419	DISTAB	JSR BCDBIN	;	CONVERT BCD FR# TO BIN
2270	1947	4CBA19		JMP SETZ		
2280	194A	0A	SHFTLF	ASL A	;	..ROTATE LEFT 5 BITS
2290	194B	26D1		ROL #D1	;	SHIFT INTO HI ADDR
2300	194D	88		DEY		
2310	194E	D0FA		BNE SHFTLF		
2320	1950	D8		CLD		
2330	1951	18		CLC		
2340	1952	65D2		ADC #D2	;	ADD IN THE COLUMN CNT
2350	1954	85D0		STA #D0	;	PUT IN LOW DIST TAB PT
2360	1956	A5D1		LDA #D1		
2370	1958	0920		ORA #BRDTAB	;	SET HI ADDR
2380	195A	85D1		STA #D1		
2390	195C	A5DD		LDA #DD	;	GET BR DIST
2400	195E	91D0		STA (#D0),Y	;	PUT IN DIST TABLE
2410	1960	60		RTS		
2420	1961					
2430	1961					MOVBLK...SUB TO MOVE A BLOCK
2440	1961					OF DATA FROM ONE LOCATION (GET)
2450	1961					TO ANOTHER LOC (STORE)
2460	1961	B4E7	MOVBLK	STY #E7	;	SET X=# OF BYTES TO MOVE
2470	1963	A000		LDY #0	;	Y= START LOW OF "GET"
2480	1965	B1E7	LOOPM	LDA (#E7),Y	;	"GET" LOW DATA
2490	1967	91E9		STA (#E9),Y	;	STORE LOW DATA
2500	1969	E6E7		INC #E7	;	INC "GET"
2510	196B	E6E9		INC #E9	;	INC "STORE"
2520	196D	D002		BNE CONTD		
2530	196F	E6EA		INC #EA	;	IF #E9=0 THEN NEXT PAGE
2540	1971	CA	CONTD	DEX		
2550	1972	D0F1		BNE LOOPM		
2560	1974	60		RTS		
2570	1975					
2580	1975	A205	MROUT4	LDX #5	;	ROUTINE TO MOVE DATA4 TO C.RAM
2590	1977	A03F		LDY #3F	;	# OF BYTES TO MOVE
2600	1979	206119		JSR MOVBLK		
2610	197C	201419		JSR DACNUM	;	GET DAC#
2620	197F	A4C6		LDY #C6	;	GET DIST TO STORE DAC#
2630	1981	91E5		STA (#E5),Y	;	PUT DAC# INTO ROUTINE
2640	1983	18		CLC		
2650	1984	98		TYA		
2660	1985	6905		ADC #5	;	INC DIST TO STORE DAC#
2670	1987	85C6		STA #C6		
2680	1989	60		RTS		
2690	198A	8A	SETZ	TXA		
2700	198B	A000		LDY #000		
2710	198D	84D1		STY #D1		
2720	198F	A005		LDY #005		
2730	1991	4C4A19		JMP SHFTLF		

END

FILMED

9-83

DTIC

580	1845	6901		ADC #001		INC THEN FRAME COUNTER
590	1847	C5CC		CMP #CC		COMPARE WITH MAX FRAMES
600	1849	D003		BNE MFR		END OF MAJOR FRAME??
610	184B	4C0304		JMP STARTR		GO GET A NEW MAJOR FRAME
620	184E	4C0F04	MFR	JMP FRCNTR		GO GET A NEW MINOR FRAME
630	1851					

1270 1887 80
1280 188A
1290 188A
1300 188A
1310 188A
1320 188A
1330 188A

RTS
; ROUTINES TO MOVE THE DATA TO C. RAM
; AND TO THE UPDATE ROUTINE. THESE ROUTINES
; PUT IN THE DAC#, WD# AND INTERVALS IF NEEDED.
;

1980 1924 ADDN =#D3
1990 1924 BIN =#D4
2000 1924 A950 BCDBIN LDA #50 ; # TO BE ADDED TO BIN#
2010 1926 85D3 STA ADDN
2020 1928 8A TXA
2030 1929 290F AND #0F ; KEEP LOWER NIBBLE

2680	1989	60		RTS
2690	198A	8A	SETZ	TXA
2700	198B	A000		LDY ##00
2710	198D	B4D1		STY #D1
2720	198F	A005		LDY ##05
2730	1991	4C4A19		JMP SHFTLF

DTIC